Multiagent Systems II
Prof. Dr. Jürgen Dix

Department of Informatics
Clausthal University of Technology
SS 2012

About this lecture
This MSc course continues the BSc course MAS I. Emphasis is now put on theory, not on applications and programming MAS. We start with classical game theory as a means for decision making in agent systems and introduce equilibrium states of a MAS. We then consider task allocation, coalition formation and reliability in MAS. After introducing the framework of modal logic, we use it to formulate several logics to reason about important aspects of agent systems: temporal and strategic properties (e.g. a group of agents has the power to bring about certain properties).

My thanks go to Dr. Nils Bulling who helped to transform my former MAS course into this new and advanced format.

Main References
Lecture Overview

Part I: Decision Making:  
*Game theory and Social Choice*
2 Chapters, 7.5 lectures

Part II: Tasks, Coalitions and Reliability
1 Chapter, 1.5 lectures

Part III: Logics for MAS
4 Chapters, 9 lectures

Exercises: 6 exercise classes (roughly fortnightly)

Outline

1 Decision Making: Games

1. Decision Making: Games
   - Examples and Terminology
   - Normal Form Games
   - Extensive Form Games
   - Repeated Games
   - Bayesian Games
   - An Example from Economics
   - References

Outline (1)

We illustrate the difference between classical AI and MAS. We present several evaluation criteria for comparing protocols.

We then introduce the formal machinery of game theory
- normal form (NF) games, where
- extensive form (tree form) games, where the history plays a role and players come up with strategies depending on the past.
- We distinguish between perfect and imperfect recall.

Important results are the minmax theorem of von Neumann (1928) and its generalization to Nash’s theorem (1950).
Outline (2)

What happens if a game is not played once, but several times or infinitely often?
- We consider repeated games, like bargaining mechanisms, and Axelrod’s observations on the iterated prisoners dilemma.
- We also consider incomplete knowledge, where players are not sure about which game they are playing: Bayes-Nash games.
- Finally we consider the existence of equilibria for market mechanisms.

Classical DAI: System Designer fixes an Interaction-Protocol which is uniform for all agents. The designer also fixes a strategy for each agent.

Outcome

What is the outcome, assuming that the protocol is followed and the agents follow the strategies?

MAI: Interaction-Protocol is given. Each agent determines its own strategy (maximising its own good, via a utility function, without looking at the global task).

Global optimum

What is the outcome, given a protocol that guarantees that each agent’s desired local strategy is the best one (and is therefore chosen by the agent)?

1.1 Examples and Terminology
We need to compare protocols. Each such protocol leads to a solution. So we determine how good these solutions are.

**Social Welfare:** Sum of all utilities

**Pareto Efficiency:** A solution $x$ is Pareto-optimal, if

there is no solution $x'$ with:

1. \( \exists \text{ agent } ag : ut_{ag}(x') > ut_{ag}(x) \)
2. \( \forall \text{ agents } ag' : ut_{ag'}(x') \geq ut_{ag'}(x) \).

**Individual rational:** The payoff should be higher than not participating at all.

### Example 1.1 (Prisoners Dilemma, Type 1)

Two prisoners are suspected of a crime (which they both committed). They can choose to (1) cooperate with each other (not confessing to the crime) or (2) defect (giving evidence that the other was involved). Both cooperating (not confessing) gives them a shorter prison term than both defecting. But if only one of them defects (the betrayer), the other gets maximal prison term. The betrayer then has maximal payoff.

<table>
<thead>
<tr>
<th></th>
<th>Prisioner 2</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Prisioner 1 cooperate</td>
<td>(3,3)</td>
<td>(5,0)</td>
</tr>
<tr>
<td>Prisioner 1 defect</td>
<td>(5,0)</td>
<td>(1,1)</td>
</tr>
</tbody>
</table>

### Stability:

**Case 1:** Strategy of an agent depends on the others. The profile \( s_A = (s_1^*, s_2^*, \ldots, s_{|A|}^*) \) is called a Nash-equilibrium, iff \( \forall i : s_i^* \) is the best strategy for agent \( i \) if all the others choose \( (s_1^*, s_2^*, \ldots, s_{i-1}^*, s_{i+1}^*, \ldots, s_{|A|}^*) \).

**Case 2:** Strategy of an agent does not depend on the others. Such strategies are called **dominant**.

- **Social Welfare:** Both cooperate,
- **Pareto-Efficiency:** All are Pareto optimal, except when both defect.
- **Dominant Strategy:** Both defect.
- **Nash Equilibrium:** Both defect.
1 Decision Making: Games

1.1 Examples and Terminology

**Prisoners dilemma revisited:** $c \geq a \geq d \geq b$

<table>
<thead>
<tr>
<th>Player 1</th>
<th>cooperate</th>
<th>defect</th>
</tr>
</thead>
<tbody>
<tr>
<td>cooperate</td>
<td>(a,a)</td>
<td>(b,c)</td>
</tr>
</tbody>
</table>

**Example 1.2 (Trivial mixed-motive, Type 0)**

<table>
<thead>
<tr>
<th>Player 2</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Player 1</td>
<td>(4,4)</td>
<td>(2,3)</td>
</tr>
<tr>
<td>C</td>
<td>(3,2)</td>
<td>(1,1)</td>
</tr>
</tbody>
</table>

**Example 1.3 (Battle of the Bismarck Sea)**

In 1943 the northern half of New Guinea was controlled by the Japanese, the southern half by the allies. The Japanese wanted to reinforce their troops. This could happen using two different routes: (1) north (rain and bad visibility) or (2) south (weather ok). Trip should take 3 days.

The allies want to bomb the convoy as long as possible. If they search north, they can bomb 2 days (independently of the route taken by the Japanese). If they go south, they can bomb 3 days if the Japanese go south too, and only 1 day, if the Japanese go north.

*Allies: What is the largest of all row minima?*  
*Japanese: What is smallest of the column maxima?*

**Battle of the Bismarck sea:**  
largest row minimum = smallest column maximum.  
This is called a *saddle point.*
1 Decision Making: Games

1.2 Normal Form Games

A finite \(n\)-person normal form game is a tuple \(<A, \text{Act}, O, \varrho, \mu\rangle\), where

- \(A = \{1, \ldots, i, \ldots, n\}\) is a finite set of players.
- \(\text{Act} = \langle A_1, \ldots, A_i, \ldots, A_n \rangle\) where \(A_i\) is the set of actions available to player \(i\). \(a \in \text{Act}\) is called an action profile. Elements of \(A_i\) are called pure strategies.
- \(O\) is the set of outcomes.
- \(\varrho : \text{Act} \rightarrow O\) assigns each action profile an outcome.
- \(\mu = \langle \mu_1, \ldots, \mu_i, \ldots, \mu_n \rangle\) where \(\mu_i : O \rightarrow \mathbb{R}\) is a real-valued utility (payoff) function for player \(i\).

Note that we distinguish between outcomes and utilities assigned to them. Often, one assigns utilities directly to actions.

Games can be represented graphically using an \(n\)-dimensional payoff matrix. Here is a generic picture for 2-player, 2-strategy games:

<table>
<thead>
<tr>
<th></th>
<th>Player 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a_1)</td>
<td>(\mu_1(a_1, a_2), \mu_2(a_1, a_2))</td>
</tr>
<tr>
<td>(a_2)</td>
<td>(\mu_1(a_1, a_2), \mu_2(a_1, a_2))</td>
</tr>
</tbody>
</table>

We often forget about \(\varrho\) (thus we are making no distinction between actions and outcomes). Thus we simply write \(\mu_i(a_1, a_2)\) instead of the more precise \(\mu_i(\varrho(a_1, a_2))\). However, there are situations where we need to distinguish between the two, in particular when talking about mechanism design (in Chapter 2, Section 3) and auctions (in Chapter 2, Section 4).
### Definition 1.5 (Common Payoff Game)

A common payoff game (team game) is a game in which for all action profiles \( a \in A_1 \times \ldots \times A_n \) and any two agents \( i, j \) the following holds: \( \mu_i(a) = \mu_j(a) \).

In such games agents have no conflicting interests. Their graphical depiction is simpler than above (the second component is not needed).

### Definition 1.6 (Constant Sum Game)

A 2-player \( n \)-strategy normal form game is called constant sum game, if there exists a constant \( c \) such that for each action profile \( a \in A_1 \times A_2 \): \( \mu_1(a) + \mu_2(a) = c \).

We usually set wlog \( c = 0 \) (zero sum games).

### Pure vs. mixed strategies

What we are really after are strategies.

#### Definition 1.7 (Pure strategy)

A pure strategy for a player is a particular action that is chosen and then played constantly.

A pure strategy profile is just an action profile \( \langle a_1, \ldots, a_n \rangle \).

Are pure strategy profiles sufficient?
Example 1.8 (Rochambeau Game)

Also known as paper, rock and scissors: paper covers rock, rock smashes scissors, scissors cut paper.

<table>
<thead>
<tr>
<th></th>
<th>P</th>
<th>S</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min</td>
<td>0</td>
<td>-1</td>
<td>1</td>
</tr>
<tr>
<td>Max</td>
<td>1</td>
<td>0</td>
<td>-1</td>
</tr>
</tbody>
</table>

What about pure vs mixed strategies?

The support of a mixed strategy is the set of actions that are assigned non-zero probabilities.

What is the payoff of such strategies? We have to take into account the probability with which an action is chosen. This leads to the expected utility $\mu_{\text{expected}}$.

Definition 1.9 (Mixed Strategy for NF Games)

Let $\langle A, \text{Act}, O, \varrho, u \rangle$ be normal form game. For a set $X$ let $\Pi(X)$ be the set of all probability distributions over $X$. The set of mixed strategies for player $i$ is the set $S_i = \Pi(A_i)$. The set of mixed strategy profiles is $S_1 \times \ldots \times S_n$. This is also called the strategy space of the game.

Note: Some books use $S_i$ to denote the set of pure strategies, and $\Sigma_i$ to denote the set of mixed strategies for player $i$.

Definition 1.10 (Expected Utility for player $i$)

The expected utility for player $i$ of the mixed strategy profile $(s_1, \ldots, s_n)$ is defined as

$$
\mu_{\text{expected}}(s_1, \ldots, s_n) = \sum_{a \in A} \mu_i(\varrho(a)) \prod_{j=1}^n s_j(a_j).
$$

What is the optimal strategy (maximising the expected payoff) for an agent in an 2-agent setting?
Example 1.11 (Fighters and Bombers)
Consider fighter pilots in WW II. A good strategy to attack bombers is to swoop down from the sun: *Hun-in-the-sun strategy*. But the bomber pilots can put on their sunglasses and stare into the sun to watch the fighters. So another strategy is to attack them from below *Ezak-Imak strategy*: if they are not spotted, it is fine, if they are, it is fatal for them (they are much slower when climbing). The table contains the survival probabilities of the fighter pilot.

<table>
<thead>
<tr>
<th>Bomber Crew</th>
<th>Look Up</th>
<th>Look Down</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fighter Pilots</td>
<td>Hun-in-the-Sun</td>
<td>0.95</td>
</tr>
<tr>
<td></td>
<td>Ezak-Imak</td>
<td>1</td>
</tr>
</tbody>
</table>

Example 1.12 (Battle of the Sexes, Type 2)
Married couple looks for evening entertainment. They prefer to go out together, but have different views about what to do (say going to the theatre and eating in a gourmet restaurant).

<table>
<thead>
<tr>
<th></th>
<th>Wife</th>
</tr>
</thead>
<tbody>
<tr>
<td>Theatre</td>
<td>(4,3)</td>
</tr>
<tr>
<td>Restaurant</td>
<td>(1,1)</td>
</tr>
</tbody>
</table>

Example 1.13 (Leader Game, Type 3)
Two drivers attempt to enter a busy stream of traffic. When the cross traffic clears, each one has to decide whether to concede the right of way of the other (C) or drive into the gap (D). If both decide for C, they are delayed. If both decide for D there may be a collision.

<table>
<thead>
<tr>
<th>Driver 2</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Driver 1</td>
<td>C</td>
<td>(2,2)</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>(4,3)</td>
</tr>
</tbody>
</table>

Example 1.14 (Matching Pennies Game)
Two players display one side of a penny (head or tails). Player 1 wins the penny if they display the same, player 2 wins otherwise.

<table>
<thead>
<tr>
<th>Player 2</th>
<th>Player 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Head</td>
<td>Tails</td>
</tr>
<tr>
<td>(1, -1)</td>
<td>(-1, 1)</td>
</tr>
<tr>
<td>(-1, 1)</td>
<td>(1, -1)</td>
</tr>
</tbody>
</table>
### Definition 1.15 (Maxmin strategy)

Given a game \( \langle \{1, 2\}, \{A_1, A_2\}, \{\mu_1, \mu_2\} \rangle \), the maxmin strategy of player \( i \) is a mixed strategy that maximizes the guaranteed payoff of player \( i \), no matter what the other player \(-i\) does:

\[
\arg\max_{s_i} \min_{s_{-i}} \mathbb{E}(s_i, s_{-i})
\]

The maxmin value for player \( i \) is \( \max_{s_i} \min_{s_{-i}} \mathbb{E}(s_i, s_{-i}) \).

The minmax strategy for player \( i \) is

\[
\arg\min_{s_i} \max_{s_{-i}} \mathbb{E}(s_i, s_{-i})
\]

and its minmax value is \( \min_{s_i} \max_{s_{-i}} \mathbb{E}(s_i, s_{-i}) \).

---

### Lemma 1.16

In each finite normal form 2-person game (not necessarily constant sum), the maxmin value of one player is never strictly greater than the minmax value for the other.

---

We illustrate the maxmin strategy using a 2-person 3-strategy constant sum game:

<table>
<thead>
<tr>
<th></th>
<th>Player B</th>
</tr>
</thead>
<tbody>
<tr>
<td>B-I</td>
<td>0</td>
</tr>
<tr>
<td>B-II</td>
<td>\frac{3}{4}</td>
</tr>
<tr>
<td>B-III</td>
<td>\frac{1}{2}</td>
</tr>
</tbody>
</table>

We assume Player A’s optimal strategy is to play strategy

- A-I with probability \( x \) and
- A-II with probability \( 1 - x \).

In the following we want to determine \( x \).

Thus Player A’s expected utility is as follows:

1. when playing against B-I: \( 0x + 1(1 - x) = 1 - x \),
2. when playing against B-II: \( \frac{5}{6}x + \frac{1}{2}(1 - x) = \frac{1}{2} + \frac{1}{3}x \),
3. when playing against B-III: \( \frac{1}{2}x + \frac{3}{4}(1 - x) = \frac{3}{4} - \frac{1}{4}x \).

This can be illustrated with the following picture (see blackboard). Thus B-III does not play any role.

Thus the maxmin point is determined by setting

\[
1 - x = \frac{1}{2} + \frac{1}{3}x,
\]

which gives \( x = \frac{3}{8} \). The value of the game is \( \frac{5}{8} \).

The strategy for Player B is to choose B-I with probability \( \frac{1}{4} \) and B-II with probability \( \frac{3}{4} \). 

---
More in accordance with the minmax strategy let us compute

\[
\arg\max_{s_i} \min_{s_{-i}} \mu_{\text{expected}}(s_1, s_2)
\]

We assume Player A plays (as above) A-I with probability \(x\) and A-II with probability \(1 - x\) (strategy \(s_1\)). Similarly, Player B plays B-I with probability \(y\) and B-II with probability \(1 - y\) (strategy \(s_2\)).

We compute \(\mu_{\text{expected}}(s_1, s_2)\)

\[
0 \cdot x \cdot y + \frac{5}{6} x (1 - y) + 1 \cdot (1 - x) y + \frac{1}{2} (1 - x) (1 - y)
\]

thus

\[
\mu_{\text{expected}}(s_1, s_2) = y\left(-\frac{4}{3} x + \frac{1}{2}\right) + \frac{1}{3} x + \frac{1}{2}
\]

According to the minmax strategy, we have to choose \(x\) such that the minimal values of the above term are maximal. For each value of \(x\) the above is a straight line with some gradient. Thus we get the maximum when the line does not slope at all!

Thus \(x = \frac{3}{8}\). A similar reasoning gives \(y = \frac{1}{4}\).

---

**Theorem 1.17 (von Neumann (1928))**

In any finite 2-person constant-sum game the following holds:

1. The maxmin value for one player is equal to the minmax value for the other. The maxmin of player 1 is usually called value of the game.
2. For each player, the set of maxmin strategies coincides with the set of minmax strategies.
3. The maxmin strategies are optimal: if one player does not play a maxmin strategy, then its payoff (expected utility) goes down.

From now on we use just \(\mu_1(s_1, s_2)\) instead of the more precise \(\mu_1^{\text{expected}}(s_1, s_2)\). It will be clear from context whether the argument is a profile (and thus it is the expected utility \(\mu^{\text{expected}}\)) or it is the utility of an outcome (and thus it is defined in the underlying game with \(\mu\)).

What is the optimal strategy (maximising the expected payoff) for an agent in an \(n\)-agent setting?
1 Decision Making: Games

1.2 Normal Form Games

Definition 1.18 (Best Response to a Profile)

Given a strategy profile

\[ s_{-i} = (s_1, s_2, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n), \]

a best response of player \( i \) to \( s_{-i} \) is any mixed strategy \( s_i^* \in S_i \) such that

\[ \mu_i(s_i^*, s_{-i}) \geq \mu_i(s_i, s_{-i}) \]

for all strategies \( s_i \in S_i \).

Is a best response unique?

Example 1.19 (Responses for Rochambeau)

How does the set of best responses look like?

1. Player 2 plays the pure strategy paper.
2. Player 2 plays paper with probability .5 and scissors with probability .5.
3. Player 2 plays paper with probability \( \frac{1}{3} \) and scissors with probability \( \frac{1}{3} \) and rock with probability \( \frac{1}{3} \).

Is a non-pure strategy in the best response set (say a strategy \( (s_1, s_2) \) with probabilities \( \langle p, 1 - p \rangle, p \neq 0 \)), then so are all other mixed strategies with probabilities \( \langle p', 1 - p' \rangle \) where \( p \neq p' \neq 0 \).

Consider the set of best responses.

Either this set is a singleton (namely when it consists of a pure strategy), or

the set is infinite.
Definition 1.20 (Nash Equilibrium (NE))

A strategy profile \( s^* = (s^*_1, s^*_2, \ldots, s^*_n) \) is a Nash equilibrium if for any agent \( i \), \( s^*_i \) is a best response to \( s^*_{-i} = (s^*_1, s^*_2, \ldots, s^*_i-1, s^*_{i+1}, \ldots, s^*_n) \).

What are the Nash equilibria in the Battle of sexes? What about the matching pennies?

Example 1.21 (Cuban Missile Crisis, Type 4)

This relates to the well-known crisis in October 1962.

<table>
<thead>
<tr>
<th></th>
<th><strong>USSR</strong></th>
<th><strong>Maintain</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>U. S.</strong></td>
<td>Withdrawal</td>
<td>(3, 3)</td>
</tr>
<tr>
<td></td>
<td>Blockade</td>
<td>(2, 4)</td>
</tr>
<tr>
<td>Air strike</td>
<td>Compromise</td>
<td>USSR victory</td>
</tr>
<tr>
<td>U. S. victory</td>
<td>Air strike</td>
<td>Nuclear War</td>
</tr>
</tbody>
</table>

Theorem 1.22 (Nash (1950))

Every finite normal form game has a Nash equilibrium.

Corollary 1.23 (Nash implies maxmin)

In any finite normal form 2-person constant-sum game, the Nash equilibria are exactly all pairs \( (s_1, s_2) \) of maxmin strategies \( (s_1 \text{ for player 1, } s_2 \text{ for player 2}) \). All Nash equilibria have the same payoff (expected utility): the value of the game, that player 1 gets.
Proof

We use Kakatuni’s theorem: Let $X$ be a nonempty subset of $n$-dimensional Euclidean space, and $f : X \rightarrow 2^X$. The following are sufficient conditions for $f$ to have a fixed point (i.e. an $x^* \in X$ with $x^* \in f(x^*)$):

1. $X$ is compact: any sequence in $X$ has a limit in $X$.
2. $X$ is convex: $x, y \in X, \alpha \in [0, 1] \Rightarrow \alpha x + (1 - \alpha) y \in X$.
3. $\forall x : f(x)$ is nonempty and convex.
4. For any sequence of pairs $(x_i, x_i^*)$ such that $x_i, x_i^* \in X$ and $x_i^* \in f(x_i)$, if $\lim_{i \to \infty} (x_i, x_i^*) = (x, x^*)$ then $x^* \in f(x)$.

Theorem 1.24 (Brouwer’s Fixpoint Theorem)

Let $D$ be the unit Euclidean ball in $\mathbb{R}^n$ and let $f : D \rightarrow D$ be a continuous mapping. Then there exists a fixed point of $f$: there is an $x \in D$ with $f(x) = x$.

Proof

Reduction to the $C^1$-differentiable case: Let $r : \mathbb{R}^n \rightarrow \mathbb{R}$ be the function $r(x) = r(x_1, x_2, \ldots, x_n) = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$. Let $\phi : \mathbb{R}^n \rightarrow \mathbb{R}$ given by:

$$\phi(x) = a_n (1/4 r(x)^4 - 1/2 r(x)^2 + 1/4)$$

on $D$ and equal to 0 in the complement of $D$, where the constant $a_n$ is chosen such that the integral of $\phi$ over $\mathbb{R}^n$ equals 1.
Consider agents 1, 2, and three outcomes A, B, C. Agents vote simultaneously for one outcome (no abstaining). The outcome with most votes wins. If there is no majority, then A is selected. The payoff functions are as follows:

\[ \mu_1(A) = \mu_2(B) = \mu_3(C) = 2, \mu_1(B) = \mu_2(C) = \mu_3(A) = 1 \text{ and } \mu_1(C) = \mu_2(A) = \mu_3(B) = 0. \]

What are the Nash equilibria and what are their outcomes?

There are many equilibria, \( \langle A, A, A \rangle, \langle B, B, B \rangle, \langle C, C, C \rangle, \) and \( \langle A, C, C \rangle. \) Also when agent 1 and 3 vote together for A (\( \langle A, A, A \rangle \) and \( \langle A, B, A \rangle \)).
It is obvious that there is not always a strategy that is strictly dominating all others (this is why the Nash equilibrium has been introduced).

Reducing games

However, often games can be reduced and the computation of the equilibrium simplified.

A rational player would never choose a strategy that is strictly dominated.

**Definition 1.27 (Dominating Strategy: Weakly, Strictly)**

A pure strategy $a_i$ is strictly dominated for an agent $i$, if there exists some other (mixed) strategy $s'_i$ that strictly dominates it, i.e. for all profiles $a_{-i} = \langle a_1, \ldots, a_{i-1}, a_{i+1} \ldots, a_n \rangle$, we have

$$\mu_i(s'_i, a_{-i}) \not\geq \mu_i(a_i, a_{-i}).$$

We say that a pure strategy $a_i$ is weakly dominated for an agent $i$, if in the above inequality we have $\geq$ instead of $\not\geq$ and the inequality is strict for at least one of the other $a_{-i}$.

**Definition 1.28 (Reduced Sets $A_{i}^\infty$, $S_{i}^\infty$)**

For an arbitrary normal form game with $A_i$ the set of pure strategies and $S_i$ the set of mixed strategies for agent $i$, we define ($A_0^i := A_i$, $S_0^i := S_i$)

$$A_i^n := \{ a_i \in A_i^{n-1} \mid \text{there is no } s'_i \in S_i^{n-1} \text{ s.t. } \mu_i(s'_i, a_{-i}) \not\geq \mu_i(a_i, a_{-i}) \text{ for all } a_{-i} \in A_i^{n-1} \}$$

$$S_i^n := \{ s' \in S_i \mid s'(a_i) \not\geq 0 \text{ only if } a_i \in A_i^n \}$$

Finally, $A_i^\infty := \bigcap_{n=0}^\infty A_i^n$ and $S_i^\infty$ is the set of all mixed strategies $s_i$ such that there is no $s'_i$ with $\mu_i(s'_i, a_{-i}) \not\geq \mu_i(s_i, a_{-i})$ for all $a_{-i} \in A_i^\infty$.
Theorem 1.29 (Solvable by Iterated Strict Dominance)

If for a finite normal form game, the sets $A_i^\infty$ are all singletons (such a game is called solvable by iterated strict dominance), then this strategy profile is the unique Nash equilibrium.

Lemma 1.30 (Church-Rosser)

Given a 2-person normal form game. All strictly dominated columns, as well as all strictly dominated rows can be eliminated without changing the Nash equilibria (or similar solution concepts). This results in a finite series of reduced games. The final result does not depend on the order of the eliminations.

Note: the last lemma is not true for weakly dominated strategies. There, the order does matter.

Note that we eliminate only pure strategies. Such a strategy might be dominated by a mixed strategy.

<table>
<thead>
<tr>
<th></th>
<th>L</th>
<th>C</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>(3, 2)</td>
<td>(2, 1)</td>
<td>(3, 1)</td>
</tr>
<tr>
<td>M</td>
<td>(1, 1)</td>
<td>(1, 1)</td>
<td>(2, 2)</td>
</tr>
<tr>
<td>D</td>
<td>(0, 1)</td>
<td>(4, 2)</td>
<td>(0, 1)</td>
</tr>
</tbody>
</table>

1. Eliminate row M.
2. Eliminate column R.

This leads to

<table>
<thead>
<tr>
<th></th>
<th>L</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>(3, 2)</td>
<td>(2, 1)</td>
</tr>
<tr>
<td>D</td>
<td>(0, 1)</td>
<td>(4, 2)</td>
</tr>
</tbody>
</table>
Elimination of Weakly Dominated actions

We consider the normal form game

\[
\begin{array}{c|cc}
 & L & R \\
 T & (1,1) & (0,0) \\
 M & (1,1) & (2,1) \\
 B & (0,0) & (2,1) \\
\end{array}
\]

1. If we first eliminate T, and then L we get the outcome \( (2,1) \).
2. If we first eliminate B, and then R we get the outcome \( (1,1) \).

We have previously introduced normal form games (Definition 1.4 on Slide 25). This notion does not allow to deal with sequences of actions that are reactions to actions of the opponent.

Extensive form (tree form) games

Unlike games in normal form, those in extensive form do not assume that all moves between players are made simultaneously. This leads to a tree form, and allows to introduce strategies, that take into account the history of the game.

We distinguish between perfect and imperfect information games. While the former assume that the players have complete knowledge about the game, the latter do not: a player might not know exactly which node it is in.
Such games can be visualised as trees. Here is the famous “Sharing Game”.

**Example 1.32 (Sharing Game)**

The game consists of two rounds. In the first, player 1 offers a certain share (namely (1) 2 for player 1, 0 for player 2, (2) 1 for player 1, 1 for player 2, (3) 0 for player 1, 2 for player 2). Player 2 can only accept, or refuse. In the latter case, nobody gets anything.

**Strategies in extensive form games**

**Definition 1.33 (Strategies in Extensive Form Games)**

Let $\Gamma = (A, \text{Act}, H, Z, \alpha, \rho, \sigma, \mu_1, \ldots, \mu_n)$ be a finite perfect information game in extensive form.

A *strategy* for player $i$ in $\Gamma$ is any function that assigns a legal move to each history owned by $i$.

The *pure strategies* of player $i$ are the elements of $\Pi_{h \in H, \rho(h) = i} (\alpha(h))$. These are also functions: whenever player $i$ can do a move, it chooses one of the actions available. Thus we can write a pure strategy as a vector $\langle a_1, \ldots, a_r \rangle$, where $a_1, \ldots, a_r$ are $i$’s choices at the respective moves.

In the sharing game, a pure strategy for player 2 is $\langle \text{no}, \text{yes}, \text{no} \rangle$. A (better) one is $\langle \text{no}, \text{yes}, \text{yes} \rangle$.

Why don’t we introduce mixed strategies?

**Best response, Nash Equilibrium**

Note that the definitions of best response and Nash equilibrium carry over (literally) to games in extensive form.

Note that in the following we are talking only about pure strategy profiles.
What are the NE’s in the sharing game?

\[ \{1, \langle y, y, y \rangle \}, \{1, \langle n, n, n \rangle \}, \{1, \langle n, n, y \rangle \}, \{1, \langle y, n, n \rangle \}, \{1, \langle y, n, y \rangle \}, \{1, \langle y, y, n \rangle \}, \{2, \langle n, y, n \rangle \}, \{2, \langle n, y, y \rangle \}, \{3, \langle n, n, y \rangle \}, \{3, \langle n, y, n \rangle \} \] are NE’s,

\[ \{1, \langle n, y, y \rangle \} \] is not. Also

\[ \{2, \langle n, n, n \rangle \}, \{2, \langle n, n, y \rangle \}, \{3, \langle n, n, y \rangle \} \] are NE’s.

We claim that only

\[ \{1, \langle y, y, y \rangle \} \] and \[ \{2, \langle n, y, y \rangle \} \] make sense.

Finite Set of payoffs

In order to obtain a finite set of payoffs, we assume that the goods are split with finite precision represented by a rounding function \( r : \mathbb{R} \to \mathbb{R} \). So, after \( t \) rounds, the goods are in fact worth \( \langle r(\delta_1^t), r(\delta_2^t) \rangle \), respectively, and if the offer is accepted, then \( a_1 \) takes \( r(x \delta_1^t) \), and \( a_2 \) gets \( r((1 - x) \delta_2^t) \).
Each strategy profile

\[ s^x : \begin{cases} a_1 \text{ offers } \langle x, 1 - x \rangle, \text{ agrees to } \langle y, 1 - y \rangle \text{ for } y \geq x \\ a_2 \text{ offers } \langle x, 1 - x \rangle, \text{ agrees to } \langle y, 1 - y \rangle \text{ iff } 1 - y \geq 1 - x \end{cases} \]

is a NE: an agreement is reached in the first round.

Proof.

A strategy profile determines a unique path from the root \( \emptyset \) of the game to one of the terminal nodes (and hence also a single profile of payoffs). Therefore one can construct the corresponding normal form game \( NF(\Gamma) \) by enumerating all strategy profiles and filling the payoff matrix with the resulting payoffs.
Example 1.36 (Generic Game in normal form)

We consider the game in Figure 4. The pure strategies of player 1 are \{⟨A, E⟩, ⟨A, F⟩, ⟨B, E⟩, ⟨B, F⟩\}. The pure strategies of player 2 are \{C, D\}.

\[
\begin{array}{c|cc}
  & C & D \\
 1 & AE & W \\
  & AF & X \\
 2 & BE & Y \\
  & BF & Z \\
\end{array}
\]

Note that ⟨B, E⟩, ⟨B, F⟩ are pure strategies that have to be considered.

Is there a converse of Lemma 1.35?

We consider prisoner’s dilemma and try to model a game in extensive form with the same payoffs and strategy profiles.

In fact, it is not surprising that we do not succeed in the general case:

**Theorem 1.37 (Zermelo, 1913)**

*For each perfect information game in extensive form there exists a pure strategy Nash equilibrium.*

In fact, this was the reason that we do not need mixed strategies for perfect information extensive games (question on Slide ??).

We will later introduce imperfect information games (in extensive form): Slide 101.
Example 1.38 (Unintended Nash equilibria)

Consider the following game in extensive form.

![Unintended Equilibrium](image)

The game depicted in Example 1.38 has two equilibria: \(\langle A, R \rangle\) and \(\langle B, L \rangle\). The latter one is not intuitive (while the first one is).

Can we refine the notion of NE and rule out this unintended equilibrium?

This leads to the notion of subgame perfect Nash equilibria:

**Definition 1.39 (Subgame Perfect NE (SPE))**

Let \(\Gamma\) be a perfect information game in extensive form.

**Subgame:** A subgame of \(G\) rooted at node \(h\) is the restriction of \(\Gamma\) to the descendants of \(h\).

**SPE:** The subgame perfect Nash equilibria (SPE) of a perfect information game \(\Gamma\) in extensive form are those Nash equilibria of \(\Gamma\), that are also Nash equilibria for all subgames \(\Gamma'\) of \(\Gamma\).

- What are the SPE’s in the Sharing game (Example 1.32)?
- What are the SPE’s in the following instance of the generic game:

<table>
<thead>
<tr>
<th></th>
<th>(C)</th>
<th>(D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(AE)</td>
<td>(2,0)</td>
<td>(1,1)</td>
</tr>
<tr>
<td>(AF)</td>
<td>(0,2)</td>
<td>(1,1)</td>
</tr>
<tr>
<td>(BE)</td>
<td>(3,3)</td>
<td>(3,3)</td>
</tr>
<tr>
<td>(BF)</td>
<td>(3,3)</td>
<td>(3,3)</td>
</tr>
</tbody>
</table>
Theorem 1.40 (Existence of SPE (Kuhn))

For each finite perfect information game in extensive form there exists a SPE.

The proof (on blackboard) is by induction on the length of histories. The SPE is therefore defined constructively.

Example 1.41 (Centipede Game)

This is a two person game which illustrates that even the notion of SPE can be critical.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>1</th>
<th>2</th>
<th>1</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>D</td>
<td>D</td>
<td>A</td>
<td>D</td>
<td>A</td>
<td>(3,5)</td>
</tr>
<tr>
<td>(1,0)</td>
<td>(0,2)</td>
<td>(3,1)</td>
<td>(2,4)</td>
<td>(4,3)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Centipede Revisited

- The Centipede game has just one SPE: All players always choose \( D \).
- This is rational, but humans often do not behave like that.
- Experiments show, that humans start with going across and do a down only towards the end of the game.

Bargaining Revisited

What about SPE’s in the bargaining game (Example 1.34)? Because of the finite precision, there is a minimal round \( T \) with \( r(\delta_i^{T+1}) = 0 \) for \( i = 1 \) or \( i = 2 \). For simplicity, assume that \( i = 2 \) and agent \( a_1 \) is the offerer in \( T \) (i.e., \( T \) is even). Then, the only SPE is given by the strategy profile \( s^\kappa \) with \( \kappa = (1 - \delta_2) \frac{1 - (\delta_1 \delta_2) \frac{T}{2}}{1 - \delta_1 \delta_2} + (\delta_1 \delta_2) \frac{T}{2} \). The goods are split \( \langle \kappa, 1 - \kappa \rangle \); the agreement is reached in the first round.
Imperfect Information

- In an extensive game with perfect information, the player does know all previous moves (and also the payoffs that result).
- In an extensive game with imperfect information, a player might not be completely informed about the past history.
- Or some moves in the past may have been done randomly, so in the future, even under the same circumstances, other action may be taken.

The Idea

- An extensive game is nothing else than a tree. Thus each node is unique and carries with it the path from the root (the history that lead to it).
- In order to model that a player does not perfectly know the past events, we introduce an equivalence relation on the nodes. That two nodes are equivalent, means that the player cannot distinguish between them.
- All nodes in one equivalence class must be assigned the same actions: otherwise the player could distinguish them.

Definition 1.42 (Information set \( I_i \), Partition)

For a set \( W \) (nodes, worlds, games) and a set of agents \( \mathcal{A} \), we define a partition \( I_i \) of agent \( i \in \mathcal{A} \) over it (or the information set of \( i \)) as an equivalence relation over \( W \). Its classes \( I_{ij} \) are also called partition classes. Thus the following holds.

A partition \( I_i \) is a set of subsets \( W_{i1}, \ldots, W_{is} \) of \( W \) such that:
1. \( \bigcup_j W_{ij} = W \) and 2. \( W_{ij} \cap W_{ij'} = \emptyset \) for \( j \neq j' \).

Definition 1.43 (Extensive Games, Imperfect Inf.)

A finite imperfect information game in extensive form is a tuple \( G = < \mathcal{A}, \text{Act}, H, Z, \alpha, \rho, \sigma, \mu_1, \ldots, \mu_n, I_1, \ldots, I_n > \) where

- \( \langle \mathcal{A}, \text{Act}, H, Z, \alpha, \rho, \sigma, \mu_1, \ldots, \mu_n \rangle \) is a perfect information game in the sense of Definition 1.31 on Slide 73,

- \( I_i \) are partitions on \( \{ h \in H : \rho(h) = i \} \) such that \( h, h' \in I_{ij} \) implies \( \alpha(h) = \alpha(h') \).
1 Decision Making: Games
1.3 Extensive Form Games

Example 1.44

Player 1 can not distinguish between the nodes connected by a dotted line.
Therefore player 1 can not play the right move.
It could play a mixed strategy: with probability $\frac{1}{2}$ choose $l$.

Now we need mixed strategies, to deal with the uncertainty.

Definition 1.45 (Pure strategy in Extensive Form)

Given an imperfect information game in extensive form, a pure strategy for player $i$ is a vector $(a_1, \ldots, a_k)$ with $a_j \in \alpha(I_{i,j})$ where $I_{i1}, \ldots, I_{ik}$ are the $k$ equivalence classes for agent $i$. Note that this vector is nothing than a function assigning an action to each node owned by player $i$.

Can we model prisoner's dilemma as an extensive game with imperfect information?

There is a pure strategy Nash equilibrium.
But we could have chosen to switch player 1 with player 2.
1 Decision Making: Games
1.3 Extensive Form Games

**NF game ↔ Imperfect game**

For pure strategies we have the following:

- Each game in normal form can be transformed into an imperfect information game in extensive form (but this is not one to one).
- Each imperfect information game in extensive form can be transformed into a game in normal form (this is one to one).

What are mixed strategies for an imperfect information game?

---

**Mixed Strategy: First try.**

- Given an imperfect information game in extensive form \( \Gamma \).
- Assign the normal form game (for any \( i \)) as usual, by enumerating the pure strategies.
- Now we can take the usual set of mixed strategies in the normal form game as the set of mixed strategies of the original game \( \Gamma \).

---

**Behavioral Strategy: Second try.**

- We consider the game from Figure 4 on Slide 89.
- Consider the following strategy for player 1. \( A \) is chosen with probability .7, \( B \) with .3 and \( E \) with probability .4 and \( F \) with .6. Such strategies are called behavioral: at each node, the (probabilistic) choice is made independently from the other nodes.
- Consider the following mixed strategy for player 1. \( \langle A, E \rangle \) is chosen with probability .6 and \( \langle B, F \rangle \) with probability .4. Thus, here we have a strong correlation: \( \langle A, F \rangle \) is not possible!

---

**Mixed vs. Behavioral**

**Definition 1.46 (Mixed and Behavioral strategies)**

Let \( G = \langle A, \text{Act}, H, Z, \alpha, \sigma, \mu_1, \ldots, \mu_n, I_1, \ldots, I_n \rangle \) be an imperfect information game in extensive form.

- **Mixed**: A mixed strategy of player \( i \) is one single probability distribution over \( i \)'s pure strategies.
- **Behavioral**: A behavioral strategy of player \( i \) is a vector of probability distributions \( P(I_{ij}) \) over the set of actions \( \alpha(I_{ij}) \) for each node \( h \). We denote by \( P(I_{ij}, h)(a) \) the probability for the action \( a \) to be taken at node \( h \).
Mixed vs. Behavioral (2)

The main difference is that for behavioral strategies, at each node the probability distribution is started freshly. Even if a player ends up in the same partition, she can choose independently of her previous choice. Whereas for mixed strategies, this choice is not independent: there is just one single distribution that relates the possible choices.

Are behavioral strategies more general?

We consider a one player game. At the start node, the player can choose $L$ or $R$. There result two nodes which can not be distinguished by the player. Again, $L$ or $R$ can be played and result in the four outcomes $o_1, o_2, o_3, o_4$.

- What is the outcome of the mixed strategy $\langle \frac{1}{2}LL, \frac{1}{2}RR \rangle$?
- It is $\langle \frac{1}{2}, 0, 0, \frac{1}{2} \rangle$
- No behavioral strategy results in this distribution.
- Therefore mixed strategies are not necessarily behavioral.

Example 1.47 (A game of imperfect recall)

We consider the following game

For mixed strategies, $\langle R, D \rangle$ is the unique NE. But for behavioral strategies, the following mixed strategy is a better response of player 1 to $D$: $\langle \frac{98}{198}L, \frac{100}{198}R \rangle$.

Are mixed strategies more general? (2)

- For mixed strategies, once decided, the pure strategy is consistently chosen. Therefore the outcome $\langle 100, 100 \rangle$ is not reachable.
- This is not true for behavioral strategies, where at each node, the probabilistic choice is done independently.
- What is the best response of player 1 to $D$ in behavioral strategies? Consider a mixed behavioral strategy: choose $L$ with probability $p$ and $R$ with $1 - p$. A little computation shows that the maximal payoff is obtained for $p = \frac{98}{198}$. 

Behavioral vs. mixed strategies?

- We have just seen that there are mixed strategies for which there are no behavioral strategies with the same outcome and vice versa.
- Therefore we introduce two concepts of Nash equilibria on the next page.
- Is there a class of games where both concepts are equivalent?

Definition 1.48 (NE for Mixed/Behavioral Strategies)

Let $G$ be an extensive game. A NE in mixed strategies is a mixed strategy profile $s^* = \langle s_1^*, s_2^*, \ldots, s_n^* \rangle$ s.t. for any agent $i$:

$$\mu_i(\langle s_i^*, s_{-i}^* \rangle) \geq \mu_i(\langle s_i, s_{-i}^* \rangle)$$

for all mixed strategies of player $i$.

A NE in behavioral strategies is a behavioral strategy profile $s^* = \langle s_1^*, s_2^*, \ldots, s_n^* \rangle$, s.t. for any agent $i$:

$$\mu_i(\langle s_i^*, s_{-i}^* \rangle) \geq \mu_i(\langle s_i, s_{-i}^* \rangle)$$

for all behavioral strategies of player $i$.

Definition 1.49 (Perfect Recall)

Let $\Gamma$ be an imperfect information game in extensive form. We say that player $i$ has perfect recall in $\Gamma$, if the following holds. If $h, h'$ are two nodes in the same $I_{ij}$ (for a $j$), and $h_0, a_0, h_1, a_1, \ldots, h_n, a_n, h$ resp. $h'_0, a'_0, h'_1, a'_1, \ldots, h'_m, a'_m, h'$ are paths from the root of the tree to $h$ (resp. $h'$), then

1. $n = m$,
2. for all $0 \leq j \leq n$: $h_j$ and $h'_j$ are in the same partition class,
3. for all $0 \leq j \leq n$: if $\alpha(h_j) = i$ then $a_j = a'_j$.

$\Gamma$ is a game of perfect recall, if all players have perfect recall. Otherwise it is called of imperfect recall.

A few games: Which have perfect recall?
**Do they model the same situation?**

![Diagram of two extensive form games](image)

**SPE:** What about subperfect equilibria (analogue of Definition 1.39 on Slide 93 for imperfect games)?

**First try:** In each information set, we have a set of subgames (a forest). Why not asking that a strategy should be a best response in all subgames of that forest?

---

**Perfect Recall: Behavioral strategies suffice?**

**Theorem 1.50 (Behavioral = Mixed (Kuhn, 1953))**

Let \( \Gamma \) be a game of perfect recall (perfect or imperfect information). Then for any mixed strategy of agent \( i \) there is a behavioral one such that both strategies induce the same probabilities on outcomes for all fixed strategy profiles of the other agents.

**Corollary 1.51**

In a game of perfect recall, it suffices to compute the Nash equilibria based on behavioral strategies.

---

**Example 1.52 (A Game with no SPE’s)**

![Diagram of a game with no SPE](image)
Nash equilibria: (L, U) and (R, D). None of them is a **subgame perfect** Nash equilibrium.

In one subgame, U dominates D, in the other D dominates U.

But (R, D) seems to be the unique choice: both players can put themselves into the others place and reason accordingly.

Requiring that a strategy is best response to all subgames is too strong.

There are two prominent refinements of SPE’s.

- One is the **trembling hand perfect** equilibrium. It is defined for normal form games.
- The other the **sequential** equilibrium is defined for games of perfect recall.

**Definition 1.53 (Sequential Equilibrium)**

A strategy profile \( s^* = (s^*_1, s^*_2, \ldots, s^*_n) \) is a sequential equilibrium of an extensive form game \( \Gamma \) if there exist probability distributions \( \mu(h) \) for each information set \( I_i \) such that

1. \( (s^*, \mu) = \lim_{n \to \infty} (s^n, \mu^n) \) for some sequence where \( s^n \) is fully mixed (so \( \mu^n \) is uniquely determined),
2. for any \( I_i \) of agent \( i \) and any alternative strategy \( s'_i \) of agent \( i \): \( \mu_i(s^* \mid h, \mu(h)) \geq \mu_i((s'_i, s_{-i}) \mid h, \mu(h)) \).

**Theorem 1.54 (Sequential Equilibrium)**

For each imperfect information game in extensive form with perfect recall there exists a sequential equilibrium. For perfect information games, each SPE is a sequential equilibrium but not vice versa.
Often games are not just played once. They are repeated finitely often (until consensus is reached). Sometimes, infinitely repeated games can be used to define equilibria.

In repeated games, it makes sense to make its choices dependent on the previous game (or the whole history).

Axelrod’s tournament

- What is the best strategy when prisoner’s dilemma is repeatedly played?
- Tit for tat: Cooperate in the first step, and then do what the other player did in the previous step.
- It turned out, that tit for tat is not only simple and easy to calculate (only the last move is considered) but also extremely powerful.
- Several experiments have shown that.
- Even counter strategies to tit for tat are difficult to find.
Axiomatic Bargaining

We assume two agents 1, 2, each with a utility function \( \mu_i : E \to \mathbb{R} \). If the agents do not agree on a result \( e \) the fallback \( e_{\text{fallback}} \) is taken.

Example 1.55 (Sharing 1 Pound)

How to share 1 Pound? Let \( e_{\text{fallback}} = 0 \).

Agent 1 offers \( \rho \) (\( 0 < \rho < 1 \)). Agent 2 agrees!

Such deals are individually rational and each one is in Nash equilibrium!

Therefore we need axioms!

Theorem 1.56 (Unique Solution)

The four axioms above determine a unique solution. This solution is given by

\[
e^* = \arg \max_e \{(\mu_1(e) - \mu_1(e_{\text{fallback}})) \times (\mu_2(e) - \mu_2(e_{\text{fallback}}))\}.
\]
Ways out:

1. Consider infinite games.
2. Keep finite games, but:
   1. Add a discount factor $\delta$: in round $n$, only the $\delta^{n-1}$th part of the original value is available.
   2. Bargaining costs: bargaining is not for free—fees have to be paid.

Problem: Nash equilibria in sequential games might exist in the first stages, but not later.

Solution: We consider SPE’s: Nash equilibria that remain Nash equilibria in every possible subgame, see Definition 1.39 on Slide 93.

Finite Games: Suppose $\delta = 0.9$. Then the outcome depends on # rounds.

<table>
<thead>
<tr>
<th>Round</th>
<th>1’s share</th>
<th>2’s share</th>
<th>Total value</th>
<th>Offerer</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n-3$</td>
<td>0.819</td>
<td>0.181</td>
<td>$0.9^{n-4}$</td>
<td>2</td>
</tr>
<tr>
<td>$n-2$</td>
<td>0.91</td>
<td>0.09</td>
<td>$0.9^{n-3}$</td>
<td>1</td>
</tr>
<tr>
<td>$n-1$</td>
<td>0.9</td>
<td>0.1</td>
<td>$0.9^{n-2}$</td>
<td>2</td>
</tr>
<tr>
<td>$n$</td>
<td>1</td>
<td>0</td>
<td>$0.9^{n-1}$</td>
<td>1</td>
</tr>
</tbody>
</table>

Infinite Games: $\delta_1$ factor for agent 1, $\delta_2$ factor for agent 2.

Theorem 1.57 (Unique solution for infinite games)

In a discounted infinite round setting, there exists a unique subgame perfect Nash equilibrium:

1. Agent 1 gets $\frac{1-\delta_1}{1-\delta_2}$.
2. Agent 2 gets the rest.
3. Agreement is reached in the first round.
1 Decision Making: Games
1.4 Repeated Games

Proof.

Let \( \pi \) the maximum undiscounted share that agent 1 can possibly get in any subgame perfect Nash equilibrium when she is offering. We can now get back two rounds and get \( 1 - \delta_2 (1 - \delta_1 \pi) \). Setting both equal gives us the result.

The last theorem is about infinite games.

On Slide 81, we have introduced a finite precision rounding function to get a finite set of payoffs.

Obviously, this assumption makes the entire game finite.

So we get an analogue of Theorem 1.57 for finite games.

Bargaining Costs

Agent 1 pays \( c_1 \), agent 2 pays \( c_2 \).

Protocol: After each round, the roles change and the fee is subtracted (\( c_1 \) from agent 1, \( c_2 \) from agent 2). Therefore the game is finite.

Theorem 1.58

(1) \( c_1 = c_2 \): Any split is a SPE.
(2) \( c_1 < c_2 \): Only one SPE: Agent 1 gets all.
(3) \( c_1 > c_2 \): Only one SPE: Agent 1 gets \( c_2 \), agent 2 gets \( 1 - c_2 \).

Proof.

(1) is obvious. (3) follows from (2): After the first round, agent 2 is in the role of agent 1. But to reach the second round, agent 2 would have to pay \( c_2 \). So agent 2 is willing to pay agent 1 its (i.e. agent 2’s) fees. So agreement is reached in the first round (no bargaining fees.

(2) Assume agent 2 offered \( \pi \) in round \( t \). Then in \( t - 1 \) agent 1 had offered \( 1 - \pi - c_2 \). Then in \( t - 2 \) agent 2 would have offered \( \pi + c_2 - c_1 \) and kept \( 1 - \pi - (c_2 - c_1) \). So in round \( t - 2k \), agent 2 would keep \( 1 - \pi - k(c_2 - c_1) \). But this would go to \(-\infty \), so agent 2 accepts 0 upfront.
1.5 Bayesian Games

What if the players do not know the payoff?
What, if they do not even know the game they are playing?

It turns out that both cases above are essentially identical! Such games are called Bayesian games or incomplete knowledge games (do not mix up with imperfect knowledge).

Example 1.59 (Uncertainty about payoffs)

Agent 1 (firm) is about to decide whether to build a new plant. There are incurring costs for this. Agent 2 (opponent) is about to buy the firm of agent 1. But agent 2 is not sure about the incurring costs for agent 1. Buying is good for agent 2 if and only if agent 1 does not build. Agent 1 has a clear dominant strategy.

<table>
<thead>
<tr>
<th>Build</th>
<th>Buy</th>
<th>Don’t</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>⟨0, −1⟩</td>
<td>⟨2, 0⟩</td>
</tr>
<tr>
<td>Don’t</td>
<td>⟨2, 1⟩</td>
<td>⟨3, −1⟩</td>
</tr>
</tbody>
</table>

1’s costs are high 1’s costs are low

Example 1.60

Same example as before, but some payoffs are different.

<table>
<thead>
<tr>
<th>Build</th>
<th>Buy</th>
<th>Don’t</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>⟨0, −1⟩</td>
<td>⟨2, 0⟩</td>
</tr>
<tr>
<td>Don’t</td>
<td>⟨2, 1⟩</td>
<td>⟨3, 0⟩</td>
</tr>
</tbody>
</table>

1’s costs are high 1’s costs are low
**Player 1:** Let $x$ be the probability of player 1 for building (obviously, she builds only when her cost is low).

**Player 2:** Let $y$ be the probability of player 2 for buying (player 2 does know nothing about the actual costs incurring for player 1).

**Costs high or low:** Let $p$ be the probability that the building costs for player 1 are high.

**Equilibrium $\langle x, y \rangle$:** We need to find pairs $\langle x, y \rangle$ that are stable in the following sense: $x$ is best for player 1 and $y$ is best for player 2, given probability $p$.

---

**Assumptions**

**Uncertainty:** Only about the payoffs, not about the strategy spaces, number of players, actions available etc.

**Common-prior:** Agents have all sorts of beliefs about other agents, about their beliefs about other agents etc. This can be very difficult to model. We make the common-prior assumption, explained on the next slide.

---

**Analysis of the game**

- $\langle 0, 1 \rangle$ is an equilibrium, independently of $p$.
- $\langle 1, 0 \rangle$ is an equilibrium iff $p \leq 0.5$.
- $\langle \frac{1}{2(1-p)}, \frac{1}{2} \rangle$ is a (mixed) equilibrium iff $p \leq 0.5$.

---

**Common-prior assumption**

The common-prior assumption is the simplifying assumption, that the probability distribution is fixed and known to the agents in advance. In Example ??, the probability distribution underlying $p$ (whether the building costs are high or not) is uniform and known to both players.
The uncertainty assumption is not restrictive.

Suppose player 1 does not know whether her opponent has two or three actions available:

<table>
<thead>
<tr>
<th></th>
<th>L</th>
<th>R</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>(1,2)</td>
<td>(3,4)</td>
<td>(9,10)</td>
</tr>
<tr>
<td>D</td>
<td>(5,6)</td>
<td>(7,8)</td>
<td>(11,12)</td>
</tr>
</tbody>
</table>

Then we define the following padded game in such a way, that the Nash equilibria are the same and the uncertainty is only in the payoffs:

<table>
<thead>
<tr>
<th></th>
<th>L</th>
<th>R</th>
<th>S</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>(1,2)</td>
<td>(3,4)</td>
<td>(9,-100)</td>
</tr>
<tr>
<td>D</td>
<td>(5,6)</td>
<td>(7,8)</td>
<td>(11,-100)</td>
</tr>
</tbody>
</table>

**Bayesian Game: Informal**

A Bayesian game for \( n \) agents consists of (1) a set \( W \) of \( n \)-person games that differ in their payoffs, (2) a probability distribution over these games (common-prior), and (3) a set of \( n \) partitions \( I_1, \ldots, I_n \) of \( W \). \( I_i \) is the set of potential games that agent \( i \) considers possible (can’t distinguish, are equivalent for her etc.). A partition \( I_i \) is a set of subsets \( W_{i1}, \ldots, W_{is} \) of \( W \) such that: (1) \( \bigcup_j W_{ij} = W \) and (2) \( W_{ij} \cap W_{ij'} = \emptyset \) for \( j \neq j' \).

\( I_i \) is the information that agent \( i \) has about the game. \( i \) does not know which game is being played, but it considers all games in one equivalence class (one of the \( W_{ij} \)) as indistinguishable. Therefore a strategy is determined per partition class.

---

**Definition 1.61 (n-Person Bayesian Game)**

A finite \( n \)-person Bayesian game is a tuple \( \langle A, G, P, I \rangle \), where

- \( A = \{1, \ldots, i, \ldots, n\} \) is a finite set of players.
- \( G \) is a set of \( n \)-person normal form games, each of which has the same action profiles \( A = \langle A_1, \ldots, A_i, \ldots, A_n \rangle \) where \( A_i \) is the set of actions available to player \( i \).
- \( P \) is a probability distribution over the set \( G \) of all games (the set of all distributions is denoted by \( \Pi(G) \)).
- \( I = \{I_1, \ldots, I_n\} \) is a set of partitions of \( G \).
Strategies for Bayesian Games

What is a strategy for agent $i$ in a Bayesian game? Clearly, a strategy must be compatible with the information the agent has about the game, i.e. it is a function from the set of partition classes into the set of (mixed) strategies of the normal form games in $G$:

$$s_i : I_i \rightarrow \Pi(A_i).$$

**Definition 1.62 (Bayes-Nash Equilibrium)**

Given a Bayesian game $(G, P, I)$ a strategy profile $s^* = (s^*_1, s^*_2, \ldots, s^*_n)$ is a Bayes-Nash equilibrium if for each agent $i$ the following holds

$$s^*_i \in \operatorname{argmax}_{s_i \in S_i} \sum_{r \in I_i} \sum_{g \in G} \mathbb{E}_{g,i}(s^*, s_i) P_(g|r)$$

$\mathbb{E}_{g,i}$ is the expected utility function for agent $i$ in game $g$.

What are the equilibria in the game depicted in Figure 8?

- The players have 4 different strategies: two partition classes and again 2 for the $2 \times 2$ games. Let player 1’s strategies for the underlying games be $U$ and $D$ and players 2’s strategies $L$ and $R$.
- Then they have to choose which strategies to play in their two partition classes. Thus player 1 can play $UU$, $UD$, $DU$, or $DD$: the first symbol stands for $I_{11}$, the second for $I_{12}$.
- Analogously, we get $LL$, $LR$, $RL$, or $RR$ for player 2.

**Figure 8**: A Bayesian game consisting of zero-sum games
Associated normal form game

\[
\begin{array}{cccc}
\text{LL} & \text{LR} & \text{RL} & \text{RR} \\
\text{UU} & 1.3 & 1.45 & 1.1 & 1.25 \\
\text{UD} & 1.8 & 1.65 & 1.8 & 1.65 \\
\text{DU} & 1.1 & 0.7 & 2.0 & 1.95 \\
\text{DD} & 1.5 & 1.15 & 2.8 & 2.35 \\
\end{array}
\]

**Exercise:** Show how to obtain these values.

---

Reduced game: The unique NE in Figure 157 is \(\langle UD, LR \rangle\).

Bayesian game: Therefore the Bayes-Nash Equilibrium of the original game is \(\langle U, L \rangle\).

---

Another Definition of Bayesian Game

- We can also define a Bayesian game by introducing an additional agent (God, Nature) that does all the probabilistic choices beforehand.
- God is the first player and then all the original players come in.
- We have thus reduced a Bayesian game to an extensive game with imperfect information: see Figure 9.
1.6 An Example from Economics

A theory for efficiently allocating goods and resources among agents, based on market prices.

**Goods:** Given \( n > 0 \) goods \( g_1, \ldots, g_n \) (coffee, mirror sites, parameters of an airplane design). We assume \( g_i \neq g_j \) for \( i \neq j \) but within each \( g_i \) everything is indistinguishable.

**Prices:** The market has prices \( p = [p_1, p_2, \ldots, p_n] \in \mathbb{R}^n \): \( p_i \) is the price of the good \( i \).
Consumers: Consumer $i$ has $\mu_i(x)$ encoding its preferences over consumption bundles $x_i = [x_{i1}, ..., x_{in}]^t$, where $x_{ig} \in \mathbb{R}^+$ is consumer $i$’s allocation of good $g$. Each consumer also has an initial endowment $e_i = [e_{i1}, ..., e_{in}]^t \in \mathbb{R}$.

Producers: Use some commodities to produce others: $y_j = [y_{j1}, ..., y_{jn}]^t$, where $y_{jg} \in \mathbb{R}$ is the amount of good $g$ that producer $j$ produces. $Y_j$ is a set of such vectors $y$.

Profit of producer $j$: $p \times y_j$, where $y_j \in Y_j$.

Profits: The profits are divided among the consumers (given predetermined proportions $\Delta_{ij}$): $\Delta_{ij}$ is the fraction of producer $j$ that consumer $i$ owns (stocks). Profits are divided according to $\Delta_{ij}$.

Definition 1.63 (General Equilibrium)

$(p^*, x^*, y^*)$ is in general equilibrium, if the following holds:

I. The markets are in equilibrium:
$$\sum_{i} x_i^* = \sum_{i} e_i + \sum_{j} y_j^*$$

II. Producer $j$ maximises profit wrt. the market
$$y_j^* = \arg \max_{y_j \in Y_j} p^* \times y_j$$

III. Consumer $i$ maximises preferences according the prices
$$x_i^* = \arg \max_{x_i \in \mathbb{R}^n \mid \text{cond}_i} \mu_i(x_i)$$

where $\text{cond}_i$ stands for
$$p^* \times x_i \leq p^* \times e_i + \sum_j \Delta_{ij} p^* \times y_j.$$
Theorem 1.64 (Pareto Efficiency)
Each general equilibrium is pareto efficient.

Theorem 1.65 (Coalition Stability)
Each general equilibrium with no producers is coalition-stable: no subgroup can increase their utilities by deviating from the equilibrium and building their own market.

Theorem 1.66 (Existence of an Equilibrium)
Let the sets $Y_j$ be closed, convex and bounded above. Let $\mu_i$ be continuous, strictly convex and strongly monotone. Assume further that at least one bundle $x_i$ is producible with only positive entries $x_{il}$.

Under these assumptions a general equilibrium exists.

Meaning of the assumptions

Formal definitions: $\rightsquigarrow$ blackboard.
Convexity of $Y_j$: Economies of scale in production do not satisfy it.
Continuity of the $\mu_i$: Not satisfied in bandwidth allocation for video conferences.
Strictly convex: Not satisfied if preference increases when she gets more of this good (drugs, alcohol, dulce de leche).

In general, there exist more than one equilibrium.

Theorem 1.67 (Uniqueness)
If the society-wide demand for each good is non-decreasing in the prices of the other goods, then a unique equilibrium exists.

Positive example: increasing price of meat forces people to eat potatoes (pasta).
Negative example: increasing price of bread implies that the butter consumption decreases.
1.7 References

Game Theory.
MIT Press.

Equilibrium points in n-person games.
Proceedings of the National Academy of Sciences of the United States of America 36, 48–49.

A Course in Game Theory.
MIT Press.

2 Decision Making: Social Choice

2. Decision Making: Social Choice

Outline (1)
We deal with voting systems and discuss
- some classical approaches, and
- an abstract framework to describe arbitrary voting mechanisms: social choice theory,
- Arrow’s theorem in this framework.
Outline (2)
A problem we have to face is that agents are lying to get what they want:

- **Tactical Voting (Mechanism Design)** is an important area in MAS. The Gibbard/Satterthwaite theorem is similar to Arrow’s theorem and deals with non-manipulable voting systems.

---

**2.1 Classical Voting Systems**

**Voting procedure**
Agents give input to a mechanism: The outcome is taken as a solution for the agents.

- **Non-ranking voting**: Each agent votes for exactly one candidate. The winner is the one with a majority of votes.
- **Approval voting**: Each agent can cast a vote for as many candidates as she wishes (at most one for each candidate). The winner is the one with a majority of votes.
- **Ranking voting**: Each agent expresses his full preference over the candidates.

---

**Definition 2.1 (Condorcet- Condition, Set)**

The **Condorcet condition**: If a candidate $x$ is chosen, then for any other candidate $y$, when just comparing $x$, $y$ the majority prefers $x$ to $y$.

The **Condorcet set** is the set of candidates that meet the Condorcet condition.

---

**Figure 12**: A Tie, but Condorcet helps.
2 Decision Making: Social Choice

2.1 Classical Voting Systems

Figure 13: Condorcet rules out all candidates.

Comparing A and B: majority for A.
Comparing A and C: majority for C.
Comparing B and C: majority for B.

Desired Preference ordering: \( A > B > C > A \)

Lemma 2.2

In approval voting, at least one of the winners is in the Condorcet set.

Definition 2.3 (Borda Protocol)

Suppose \( O \) is the number of candidates. Each agents gives his best candidate \( |O| \) points, the second best gets \( |O| - 1 \) points, etc. After all agents have voted, sum up, across all voters. The highest count wins.

Winner turns loser and loser turns winner.

<table>
<thead>
<tr>
<th>Agent</th>
<th>Preferences</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( a &gt; b &gt; c &gt; d )</td>
</tr>
<tr>
<td>2</td>
<td>( b &gt; c &gt; d &gt; a )</td>
</tr>
<tr>
<td>3</td>
<td>( c &gt; d &gt; a &gt; b )</td>
</tr>
<tr>
<td>4</td>
<td>( a &gt; b &gt; c &gt; d )</td>
</tr>
<tr>
<td>5</td>
<td>( b &gt; c &gt; d &gt; a )</td>
</tr>
<tr>
<td>6</td>
<td>( c &gt; d &gt; a &gt; b )</td>
</tr>
<tr>
<td>7</td>
<td>( a &gt; b &gt; c &gt; d )</td>
</tr>
</tbody>
</table>

Borda count without \( d \):

\( e \) wins: 20, \( b \): 19, \( a \): 18, \( d \) loses: 13

Binary protocol: Pairwise comparison.

Take any two candidates and determine the winner. The winner enters the next round, where it is compared with one of the remaining candidates.

Which ordering should we use?
2 Decision Making: Social Choice

2.1 Classical Voting Systems

Coomb’s method: Each voter ranks all candidates in linear order. If there is no candidate ranked first by a majority of all voters, the candidate which is ranked last (by a majority) is eliminated. The last remaining candidate wins.

d’Hondt’s method: Each voter cast his votes. Seats are allocated according to the quotient $\frac{V}{s+1}$ ($V$ the number of votes received, $s$ the number of seats already allocated).

Proportional Approving voting: Each voter gives points $\frac{1}{1}$, $\frac{1}{2}$, $\frac{1}{3}$, …, $\frac{1}{n}$ for his candidates (he can choose as many or few as she likes). The winning candidates are those, where the sum of all points is maximal (across all voters).

2.2 Social Choice Theory

Formal model for social choice

What is a general model for voting, based on preferences over the set of candidates? Given such a model, we are interested to define fair elections.

- A set of agents, $O$ set of possible outcomes.
  ($O$ could be $A$, a set of laws, or a set of candidates).

Voting based on total orders $\text{Ord}^\text{tot}$

The voting of agent $i$ is described by a binary relation $\prec_i \subseteq O \times O$,

which we assume to be irreflexive, transitive and total: Ties are not allowed! We denote by $\text{Ord}^\text{tot}$ the set of all such binary relations.
Often, not all subsets of $O$ are votable, only a subset $V \subseteq 2^O \setminus \emptyset$. The simplest scenario is for $V = \{ \{ o \} : o \in O \}$. Each $v \in V$ represents a possible “set of candidates”. The voting model then has to select some of the elements of $v$.

Each agent votes independently of the others. But we also allow that only a subset is considered. Let therefore be $U \subseteq \prod_{i=1}^{\vert A \vert} \text{Ord}_i^{\text{tot}}$.

The set $U$ represents the set of agents (and their preferences over the candidates) participating at the election and casting their votes.

We are now defining what a selection process, an election mechanism really is.

### Definition 2.4 (Social choice: 2 versions)

1. A **social choice function** is any function $C^*: V \times U \rightarrow O; (v, (\prec_1, \ldots, \prec_{\vert A \vert})) \mapsto o$.

2. A **social choice correspondence** is any function $W^*: V \times U \rightarrow 2^O; (v, (\prec_1, \ldots, \prec_{\vert A \vert})) \mapsto v'$.

Finally we call $C^*$ **dictatorial**, if there exists an agent $i$ such that for all $l = (\prec_1, \ldots, \prec_{\vert A \vert})$: $C^*(v, l) = \max_{i} o$.

### Theorem 2.6 (May (1952))

If there are only two candidates, there is a social choice function which is not dictatorial but yet satisfies unanimity and monotonicity.
Theorem 2.7 (Muller-Satterthwaite (1977))

If there are at least 3 candidates, then any social choice function satisfying unanimity and monotonicity must be dictatorial.

What about majority voting?

A variant of the above theorem is for correspondences. In that case, we require correspondences to satisfy the following two conditions (sometimes called prime directive):

1. $W^*(v, (≺_1, \ldots, ≺_{|A|})) \neq \emptyset$ and
2. $W^*(v, (≺_1, \ldots, ≺_{|A|})) \subseteq v$.

Such a function simply selects a subset of $v$: the elected members of the list $v$.

Independence of irrelevant alternatives (IIA): For all $v \in V$: if $(\forall i \in A : ≺_i = ≺'_{i|v|})$ then $W^*(v, (≺_1, \ldots, ≺_{|A|})) = W^*(v, (≺'_1, \ldots, ≺'_{|A|}))$.

Unanimity: If $o \in v$ and $(\forall i \in A : o' ≺_i o)$ then $o' \notin W^*(v, (≺_1, \ldots, ≺_{|A|}))$.

Consistency: If $\{o, o'\} \subseteq v$ and $W^*(v, u) \cap \{o, o'\} = \{o\}$ and $\{o, o'\} \subseteq v'$ then $o' \notin W^*(v', u)$.

Theorem 2.8 (Social choice correspondence)

Any social choice correspondence satisfying independence of irrelevant alternatives, unanimity and consistency must be dictatorial.
We are now relaxing our assumptions considerably (and thereby strengthening Arrow’s theorem on Slide 205).

**Voting based on partial orders**

The voting of agent $i$ is described by a binary relation

$$\prec_i \subseteq O \times O,$$

which we assume to be irreflexive, transitive and antisymmetric: Ties are now allowed! We denote by $\text{Ord}^{\text{par}}$ the set of all such binary relations.

A social welfare function is any function

$$f^*: U \to \text{Ord}^{\text{par}}; (\prec_1, \ldots, \prec_{|A|}) \mapsto \prec^*$$

For each $V \subseteq 2^O \setminus \{\emptyset\}$ the function $f^*$ w.r.t. $U$ induces a choice function $C_{(\prec_1, \ldots, \prec_{|A|})}$ as follows:

$$C_{(\prec_1, \ldots, \prec_{|A|})} = \text{def} \left\{ \begin{array}{l}
V \mapsto V \Rightarrow C_{(\prec_1, \ldots, \prec_{|A|})}(v) = \max_{\prec^*|V} v
\end{array} \right.$$  

max$_{\prec^*|V} v$ are the maximal elements in $v$ according to $\prec^*|V$. Each tuple $u = (\prec_1, \ldots, \prec_{|A|})$ determines the election for all possible $v \in V$.

**Definition 2.9 (Social welfare function)**

What are desirable properties for $f^*$?

- **Pareto-Efficiency**: for all $o, o' \in O$: ($\forall i \in A : o \prec^*_i o'$) implies $o \prec^* o'$.
- **Independence of Irrelevant Alternatives**: for all $o, o' \in O$: ($\forall i \in A : o \prec'_i o' \iff o \prec^*_i o'$).

Note that this implies in particular

$$\forall i \in A : \prec^*_i|v = \prec'_i|v$$

$$\Rightarrow \forall o, o' \in v, \forall v' \in V \text{ s.t. } v \subseteq v' : (o \prec^*_i|v' o' \iff o \prec^*|v' o')$$

**Majority Vote**

The simple majority vote protocol does not satisfy the Independence of irrelevant alternatives.
We consider 7 voters \( A = \{ w_1, w_2, \ldots, w_7 \} \) and \( O = \{ a, b, c, d \} \), \( V = \{ a, b, c, d \} \). The columns in the following table represent two different preference orderings of the voters (black and red).

<table>
<thead>
<tr>
<th>( \prec_1 (\prec) )</th>
<th>( \prec_2 (\prec) )</th>
<th>( \prec_3 (\prec) )</th>
<th>( \prec_4 (\prec) )</th>
<th>( \prec_5 (\prec) )</th>
<th>( \prec_6 (\prec) )</th>
<th>( \prec_7 (\prec) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1 (2)</td>
<td>1 (2)</td>
<td>1 (1)</td>
<td>1 (1)</td>
<td>2 (2)</td>
<td>2 (2)</td>
</tr>
<tr>
<td>b</td>
<td>2 (3)</td>
<td>2 (3)</td>
<td>2 (2)</td>
<td>2 (2)</td>
<td>1 (1)</td>
<td>1 (1)</td>
</tr>
<tr>
<td>c</td>
<td>3 (4)</td>
<td>3 (4)</td>
<td>3 (3)</td>
<td>3 (3)</td>
<td>3 (3)</td>
<td>3 (3)</td>
</tr>
<tr>
<td>d</td>
<td>4 (1)</td>
<td>4 (1)</td>
<td>4 (4)</td>
<td>4 (4)</td>
<td>4 (4)</td>
<td>4 (4)</td>
</tr>
</tbody>
</table>

Let \( \prec^* \) be the solution generated by the \( \prec \), and \( \prec^* \) the solution generated by the \( \prec \). Then we have for \( i = 1, \ldots, 7 \): \( b \prec \prec \prec a \) iff \( b \prec \prec \prec a \), but \( b \prec^* \prec a \) and \( a \prec^* \prec b \).

The latter holds because on the whole set \( O \), \( \prec^* \) gets selected 4 times and \( b \) only 3 times, while for \( \prec^* \) \( a \) gets selected only 2 times but \( b \) gets still selected 3 times. The former holds because we even have \( \prec_i(a,b,c) = \prec_i(a,b,c) \).

The introduction of the irrelevant (concerning the relative ordering of \( a \) and \( b \)) alternative \( d \) changes everything: the original majority of \( a \) is split and drops below one of the less preferred alternatives (\( b \)).

Before turning to Arrow’s theorem, we state the following surprising fact.

**Lemma 2.10**

Let the social welfare function satisfy **Pareto efficiency** and **Independence of Irrelevant Alternatives**. Let \( o \in O \) and suppose every agent \( i \) puts \( o \) either on top (greatest element wrt. \( \prec_i \)) or to the bottom (smallest element wrt. \( \prec_i \)). Then \( o \) is either a minimal or a maximal element of \( \prec^* \).

**Proof.**

Suppose not, i.e there is a \( a, b \in O \) (\( a \neq o, b \neq o \)) such that \( a \prec^* \prec o \prec^* b \) and thus \( a \prec b \).

Where are \( a, b \) located in the rankings \( \prec_i \)?

We construct rankings \( \prec_i' \) by moving \( a \) above \( b \) (if not yet) without disturbing the preferences between \( o, a \) and between \( o, b \). By IIA, this does not change the ranking between \( o, a \) and between \( o, b \) for \( \prec^* \), thus \( a \prec^* b \).

But the new profile satisfies \( b \prec_i' a \) for all agents \( i \), so, by pareto efficiency, \( b \prec^* a \), a contradiction.

**Theorem 2.11 (Arrow (1951))**

If the social welfare function \( f^* \) is (1) pareto efficient and (2) independent from irrelevant alternatives, then there always exists a dictator: for all \( U \subseteq \prod_{i=1}^{\vert A \vert} \text{Ord}_{\text{par}}(\text{Ord}_{\text{par}})

\[ \exists i \in A : \forall o, o' \in O : o \prec_i o' \text{ iff } o \prec^* o'. \]

To be more precise: for all \( U \subseteq \prod_{i=1}^{\vert A \vert} \text{Ord}_{\text{par}}(\text{Ord}_{\text{par}})

\[ \exists i \in A : \forall (\prec_1, \ldots, \prec_{\vert A \vert}) \in U : \forall o, o' \in O : o \prec_i o' \text{ iff } f^*((\prec_1, \ldots, \prec_{\vert A \vert})) o'. \]
Proof (of Arrows theorem).
The proof is the third proof given by John Geanakoplos (1996) and based on the following

Lemma 2.12 (Strict Neutrality)
Consider two pairs of alternatives $a, b$ and $\alpha, \beta$. Suppose each voter strictly prefers $a$ to $b$ or $b$ to $a$, i.e. for all $i$: $a \prec_i b$ or $b \prec_i a$.
Suppose further that each voter has the same preference for $\alpha, \beta$ as she has for $a, b$.
Then either $a \prec^* b$ and $\alpha \prec^* \beta$ or $b \prec^* a$ and $\beta \prec^* \alpha$ (strict preference).

By pareto efficiency, we have $a \prec^* \alpha$ (for $\alpha \neq a$) and $\beta \prec^* b$ (for $\beta \neq b$).
By IIA, we have $b \preceq^* a$. Using transitivity, we get $\beta \prec^* \alpha$.
By IIA again, we also get $\beta \prec^* \alpha$ (because $\alpha \prec^*_i \beta$ iff $\alpha \prec^*_i \beta$).

By independence of irrelevant alternatives, we can change the roles of $(a, b)$ with $(\alpha, \beta)$ and get $b \prec^* a$ (note that there is no equality sign $\preceq^*$).

□

Proof (of the lemma).
Because $|A| \geq 3$, the pair $(a, b)$ is distinct from $(\alpha, \beta)$. We assume wlog that $b \preceq^* a$. We are now constructing a different profile $\prec^*_1, \ldots, \prec^*_|A| |A|$ obtained as follows (for all $i$):
- If $a \neq \alpha$, then we change $\prec_i$ by moving $\alpha$ just above $a$.
- If $b \neq \beta$, then we change $\prec_i$ by moving $\beta$ just below $b$.
This can be done by maintaining the old preferences between $\alpha$ and $\beta$ (as preferences between $a$ and $b$ are strict).

The proof of Arrows theorem is by considering two alternatives $a, b$ and the profile where $a \prec_i b$ for all agents $i$. By pareto efficiency, $a \prec^* b$. Note that this reasoning is true for all rankings with the same relative preference between $a$ and $b$.

We now consider a sequence of profiles $\prec_1, \ldots, \prec_|A|$ (from $i = 0, \ldots, |A|$, starting with the one described above ($i = 0$)), where in step $i$, we let all agents numbered $\leq i$ change their profile by moving $a$ above $b$ (leaving all other rankings untouched).
There must be one step, let’s call it \( n^* \), where \( a \prec_{n^*-1} b \) but \( b \prec_{n^*} a \) (because of pareto efficiency and strict neutrality).

(Here we denote by \( \prec_{n^*-1} \) the ordering obtained from \( \prec_1, \ldots, \prec_{|A|} \) and, similarly, for \( \prec_{n^*} \)).

Note again that this reasoning is true not just for one profile \( \prec_1, \ldots, \prec_{|A|} \), but for all such profiles with the same relative ranking of \( a \) and \( b \).

We claim that \( n^* \) is a dictator.

We apply strict neutrality to the pair \((c, \beta)\) and \((b, a)\).

Because both pairs have the same relative ranking in \( \prec_1, \ldots, \prec_{|A|} \), we have \( \beta \prec_{n^*} c \).

We also apply strict neutrality to the pair \((\alpha, c)\) and \((a, b)\).

Because both pairs have the same relative ranking in \( \prec_1, \ldots, \prec_{|A|} \), we have \( c \prec_{n^*} \alpha \).

By transitivity: \( \beta \prec_{n^*} \alpha \).

Take any pair of alternatives \( \alpha, \beta \) and assume wlog \( \beta \prec_{n^*} \alpha \).

Take \( c \not\in \{\alpha, \beta\} \) and consider the new profile \( \prec_1, \ldots, \prec_{|A|} \) obtained as follows from \( \prec_1, \ldots, \prec_{|A|} \):

- for \( 1 \leq i \leq n^* \): we put \( c \) on top of each \( \prec_i \).
- for \( n^* + 1 \leq i \leq |A| \): we put \( c \) in between \( \alpha \) and \( \beta \).
- for \( n^* \leq i \leq |A| \): we put \( c \) to the bottom of each \( \prec_i \).

We are changing the profile \( \prec_1, \ldots, \prec_{|A|} \) by moving \( c \) around in a very particular way.

Ways out (of Arrow’s theorem):
1. Choice function is not always defined.
2. Independence of alternatives is dropped.
2.3 Mechanism Design

Lying and manipulation

- What if agents vote *tactically*? I.e. they know the voting design and do not vote *truthfully*, but such that their preferred choice is elected after all?
- Is it possible to come up with an *implementation* of a social choice function (or correspondence) that *can not be manipulated*?

Example 2.13 (Babysitting, Shoham)

You are babysitting 4 kids (Will, Liam, Vic and Ray). They can chose among (a: going to the video arcade, b: playing baseball, c: going for a leisurely car ride). The kids give their true preferences as follows:

<table>
<thead>
<tr>
<th></th>
<th>Will</th>
<th>Liam</th>
<th>Vic</th>
<th>Ray</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>b</td>
<td>b</td>
<td>a</td>
<td>c</td>
</tr>
<tr>
<td>2</td>
<td>a</td>
<td>a</td>
<td>c</td>
<td>a</td>
</tr>
<tr>
<td>3</td>
<td>c</td>
<td>c</td>
<td>b</td>
<td>b</td>
</tr>
</tbody>
</table>

Majority voting, breaking ties alphabetically.

Suppose Ray hates playing basketball but knows that his fellows do like it. *How can he vote to avoid ending up playing basketball?*

Note that the other kids do not disclose their preferences, but Ray might know them. Other kids may have other preferences, so choosing “a” is not a dominant strategy.
Definition 2.14 (Mechanism)

A mechanism \( \langle \text{Act}, M \rangle \) where

1. \( \text{Act} = A_1 \times \ldots \times A_A \) where \( A_i \) is the set of actions available to agent \( i \).
2. \( M : \text{Act} \rightarrow \Pi(O) \) where \( \Pi(O) \) is the set of all probability distributions over the set of outcomes.

This is a very general definition: it allows arbitrary actions. What about our babysitter example?

It is deterministic: Each child votes for one choice \( M((a_1, \ldots, a_A)) = 1 \). The selected outcome is the one with most votes.

Mechanism vs. Social Choice Function

What is the difference between a social choice function and a mechanism?

- Social choice function is often considered to be truthful: it is based on the real preferences.
- A mechanism is an implementation (or not!) of a social choice function.
- MD is also called inverse game theory or incentive engineering.

Definition 2.15 (Mechanism/Implementation)

Let \( \mathcal{A} \) and \( O \) be fixed. Let a social choice function \( C^* \) be given.

We say that a mechanism \( \langle \text{Act}, M \rangle \) implements the function \( C^* \) in dominant strategies, if for all utility profiles \( \mu \) the game \( \langle \mathcal{A}, \mathcal{A}, O, M, \mu \rangle \) has an equilibrium in dominant strategies and for any equilibrium \( \langle s_1^*, s_2^*, \ldots, s_n^* \rangle \) we have:

\[
M((s_1^*, s_2^*, \ldots, s_n^*)) = C^*(\mu)
\]
Implementation in dominant strategies

What about our example? Is there a mechanism implementing our social choice function in dominant strategies?

Exercise

What if we redefine the last definition in terms of Nash-equilibria? Is there a mechanism implementing our social choice function in Nash-equilibria?

Lying

There are situations, where one does not want to reveal the true preferences. Lying might pay off: not only to get the desired result, but also to ensure that critical information is not disclosed.

Definition 2.16 (Bayes-Nash Implementation)

Let $A$ and $O$ be fixed. Let $U$ be the set of all utility profiles $\mu$ over $O$. Let $G$ be the set of games $\langle A, A, O, M, \mu \rangle$ for $\mu \in U$. Let $P$ be a probability distribution on $U$ and let $I = \{I_1, \ldots, I_n\}$ be a set of partitions, one for each agent. Finally, let a social choice function $C^*$ be given.

We say that a mechanism $\langle Act, M \rangle$ implements the function $C^*$ in Bayes-Nash equilibria wrt. $P$ and $I$, if there exists a Bayes-Nash equilibrium of the Bayesian game $\langle A, G, P, I \rangle$, such that for each game $g \in G$ and each action profile $\langle a_1, \ldots, a_n \rangle \in Act$ that can arise in $g$, it holds that $M(\langle a_1, \ldots, a_n \rangle) = C^*(\mu)$.

Definition 2.17 (Direct Mechanism)

A direct mechanism wrt. a set of agents $A$ and a set of outcomes $O$ is a mechanism $\langle Act, M \rangle$ with

$$A_i = \{ \mu_i : \mu_i \text{ are utility functions} \}.$$

But: agents may lie and not reveal their true utilities.
2 Decision Making: Social Choice
2.3 Mechanism Design

Truthful or strategy-proof mechanism
A mechanism is **truthful** (or **strategy-proof**) in dominant strategies, if for any utility profile, in the resulting game it is a dominant strategy for each agent to announce its true utility function.

**Theorem 2.18 (Revelation)**

If there exists an implementation of a social choice rule in dominant strategies, then there is also a direct and truthful mechanism implementing the same function.

The same theorem is also true for implementation in Nash equilibria (with the same proof).

**Proof.**
The proof is simple: one simply builds-in the lying-part into the procedure. That is, one lets the procedure do what is best for oneself.

Relaxing our assumptions
We assume that the preferences of our agents satisfy irreflexivity, transitivity and antisymmetry: again, ties are allowed. For a preference $\prec$, we define

$$\text{top}(\prec) := \{ o \in O : \text{there is no } o' \text{ with } o \prec o' \}$$

We also define $a \npre \prec b$ iff $a$ and $b$ are incomparable wrt. $\prec$.

We call a social correspondence **strategy-proof**, if it is best for each agent to choose its preferences truthfully: deviating does not pay off.

**Gibbard/Satterthwaite**
If a social correspondence is strategy-proof and each outcome is in principle votable, then there is a dictator.

The original theorem was based on total orders. The stronger version here is due to Pini/Rossi/Venable/Walsh (AAMAS 2006).
What do we mean by each outcome is in principle votable? This condition is also called citizen sovereignty:

**Definition 2.19 (Citizen Sovereignty)**
A social correspondence

\[ W^* : \text{Ord}^{\text{par}} \to 2^O; (\prec_1, \ldots, \prec_{|A|}) \mapsto v' \]

satisfies citizen sovereignty (or is onto) if for any \( v' \subseteq O \) there is a \( (\prec_1, \ldots, \prec_{|A|}) \in \text{Ord}^{\text{par}} \) such that

\[ W^*((\prec_1, \ldots, \prec_{|A|})) = v' \]

Intuitively, strategy-proof means that the social correspondence is non-manipulable: there is no tactic voting (disclosing ones true preferences) to ensure the desired result.

**Definition 2.20 (Strategy-proof social corresp.)**
A social correspondence

\[ W^* : \text{Ord}^{\text{par}} \to 2^O; (\prec_1, \ldots, \prec_{|A|}) \mapsto v' \]

is called strategy-proof, if the following two conditions hold for every agent \( i \), for any two profiles \( p = (\prec_1, \ldots, \prec_i, \ldots, \prec_{|A|}) \) and \( p' = (\prec_1, \ldots, \prec_i', \ldots, \prec_{|A|}) \) that only differ in agent \( i \)'s preference:

1. \( \forall a \in W^*(p) \setminus W^*(p'), \forall b \in W^*(p') : \)
   - if \( a \prec_i b \) then \( a \prec_i' b \) or \( a \prec_i' b \), and
   - if \( a \prec_i' b \) then \( a \prec_i b \).

2. \( \forall a \in W^*(p) \setminus W^*(p'), \exists b \in W^*(p') \) such that:
   - if \( b \prec_i a \) then \( a \prec_i' b \) or \( a \prec_i' b \), and
   - if \( a \prec_i' b \) then \( a \prec_i b \).

Here are two examples for social correspondences.

1. Let \( W_1^* : \text{Ord}^{\text{par}} \to 2^O ; \)

\[ W_1^*((\prec_1, \ldots, \prec_{|A|})) := \bigcup_{i=1}^{|A|} \text{top}(\prec_i) \]

2. Let \( W_2^* : \text{Ord}^{\text{par}} \to 2^O ; \)

\[ W_2^*((\prec_1, \ldots, \prec_{|A|})) := \text{top}(\prec_{\text{pareto}}) \]

where \( a \prec_{\text{pareto}} b \) iff \( a \prec_i b \) for all \( 1 \leq i \leq |A| \), otherwise \( a, b \) are incomparable.
To formulate our version of the Gibbard-Satterthwaite Theorem, we need to introduce the notion of a weak dictator.

**Definition 2.21 (Weak dictator)**

An agent \( l \) is called a weak dictator for the social correspondence \( W^* \), if for all profiles \( \langle \prec_1, \ldots, \prec_{|A|} \rangle \) the following holds:

\[
W^*(\langle \prec_1, \ldots, \prec_{|A|} \rangle) \cap \text{top}(\prec_i) \neq \emptyset
\]

In Definition 2.5 on Slide 193 we have defined unanimity and monotonicty for social choice functions, assuming that the underlying order is total. Now we do it for partial orders and correspondences.

**Definition 2.22 (Unanimity of Correspondences)**

We say that a social choice correspondence \( W^* \) satisfies unanimity if the following holds:

1. For any profile \( l = \langle \prec_1, \ldots, \prec_{|A|} \rangle \) with \( o \in \text{top}(\prec_i) \) for all agents \( i \), we have \( o \in W^*(\langle \prec_1, \ldots, \prec_{|A|} \rangle) \).
2. For any profile \( l = \langle \prec_1, \ldots, \prec_{|A|} \rangle \) with \( \{o\} = \text{top}(\prec_i) \) for all agents \( i \), we have \( \{o\} = W^*(\langle \prec_1, \ldots, \prec_{|A|} \rangle) \).

**Definition 2.23 (Monotonicity of a Correspondance)**

A social choice correspondence \( W^* \) function satisfies monotonicity, if given any profiles \( l = \langle \prec_1, \ldots, \prec_{|A|} \rangle \), and \( l' = \langle \prec_1, \ldots, \prec_{|A|} \rangle \), the following holds:

1. If \( o \in W^*(\langle \prec_1, \ldots, \prec_{|A|} \rangle) \) and any \( o' \) and all \( i \)
   1. \( o' \prec_i o \) implies \( o' \prec_i o \)
   2. Then \( o \in W^*(\langle \prec_1, \ldots, \prec_{|A|} \rangle) \)
2. If \( A = W^*(\langle \prec_1, \ldots, \prec_{|A|} \rangle) \) and all \( o \in A \) and any \( o' \) and all \( i \)
   1. \( o' \prec_i o \) implies \( o' \prec_i o \)
   2. Then \( A = W^*(\langle \prec_1, \ldots, \prec_{|A|} \rangle) \).
**Definition 2.24 (Strategy-Proofness)**

A correspondence $W^*$ is called **strategy-proof**, if for every agent $i$, for every pair of profiles $l = (≺_{1}, \ldots, ≺_{i}, \ldots, ≺_{|A|})$, $l' = (≺_{1}, \ldots, ≺', \ldots, ≺_{|A|})$ that differ only for agent $i$'s ranking, it holds:

1. for all $o \in W^*(l) \setminus W^*(l')$, for all $o' \in W^*(l')$
   - $o ≺_{i} o'$ implies $o ≺_{i} o'$,
   - $o ≻_{i} o'$ implies $o ≻_{i} o'$ or $o' ≺_{i} o'$
2. for all $o \in W^*(l)$ there is a $o' \in W^*(l')$ such that
   - if $o' ≺_{i} o$ then $o ≻_{i} o'$ or $o ≺_{i} o'$.
   - $o ≻_{i} o'$ implies $o ≺_{i} o'$.

**Theorem 2.25 (Gibbard/Satterthwaite, partial order)**

Let $|O| \geq 3$, $|A| \geq 2$ and let $W^*$ be any social correspondence based on partial orderings $W^* : Ord_{par} \rightarrow 2^O$

If $W^*$ is strategy-proof and satisfies citizen-sovereignty ($W^*$ is onto), then there always exists a weak dictator.

The last definition says that a social choice function is strategy-proof, if an agent can remove an element $a$ from the set of winners only by worsening its rank with respect to at least one of the new winners, and not improving it with respect to any other new winner. It is not possible for an agent to make $a$ disappear from the set of winners by improving its ranking in her preference ordering: this would be tactical voting.

It is easy to see that the assumptions imply monotonicity and that monotonicity and citizen-sovereignty imply unanimity. The theorem then follows from Arrow's theorem on Slide 205.

**Lemma 2.26**

If a social correspondence is strategy-proof and onto, then it is unanimous and monotonic.
2.4 Auctions

While voting binds all agents, auctions are always deals between 2.

Types of auctions:
- **first-price open cry**: (English auction), as usual.
- **first-price sealed bid**: one bids without knowing the other bids.
- **dutch auction**: (descending auction) the seller lowers the price until it is taken.
- **second-price sealed bid**: (Vickrey auction) Highest bidder wins, but the price is the second highest bid!

Three different auction settings:
- **private value**: Value depends only on the bidder (cake).
- **common value**: Value depends only on other bidders (treasury bills).
- **correlated value**: Partly on own’s values, partly on others.

What is the best strategy in Vickrey auctions?

**Theorem 2.27** *(Private-value Vickrey auctions)*

The dominant strategy of a bidder in a Private-value Vickrey auction is to bid the true valuation.

Therefore it is equivalent to English auctions.

Vickrey auctions are used to:
- allocate computation resources in operating systems,
- allocate bandwidth in computer networks,
- control building heating.
Are first-price auctions better for the auctioneer than second-prize auctions?

**Theorem 2.28 (Expected Revenue)**

All 4 types of protocols produce the same expected revenue to the auctioneer (assuming (1) private value auctions, (2) values are independently distributed and (3) bidders are risk-neutral).

Why are second price auctions not so popular among humans?

1. Lying auctioneer.
2. When the results are published, subcontractors know the true valuations and what the winner saved. So they might want to share the profit.

Inefficient Allocation

Auctioning heterogeneous, interdependent items.

**Example 2.29 (Task Allocation)**

Two delivery tasks $t_1$, $t_2$. Two agents.

The global optimal solution is not reached by auctioning independently and truthful bidding.

$t_1$ goes to agent 2 (for a price of 2) and $t_2$ goes to agent 1 (for a price of 1.5).

Even if agent 2 considers (when bidding for $t_2$) that she already got $t_1$ (so she bids $\text{cost}([t_1, t_2]) - \text{cost}([t_1]) = 2.5 - 1.5 = 1$) she will get it only with a probability of 0.5.
What about full lookahead? \(\Rightarrow\) blackboard.

Therefore:
- It pays off for agent 1 to bid more for \(t_1\) (up to 1.5 more than truthful bidding).
- It does not pay off for agent 2, because agent 2 does not make a profit at \(t_2\) anyway.
- Agent 1 bids 0.5 for \(t_1\) (instead of 2), agent 2 bids 1.5. Therefore agent 1 gets it for 1.5. Agent 1 also gets \(t_2\) for 1.5.

Example 2.30 (Incentive to counterspeculate)
Suppose bidder 1 does not know the (private-) value \(v_1\) of the item to be auctioned. To determine it, she needs to invest cost. We also assume that \(v_1\) is uniformly distributed: \(v_1 \in [0, 1]\).
For bidder 2, the private value \(v_2\) of the item is fixed: \(0 \leq v_2 < \frac{1}{2}\). So his dominant strategy is to bid \(v_2\).
Should bidder 1 try to invest cost to determine his private value? How does this depend on knowing \(v_2\)?

Answer: Bidder 1 should invest cost if and only if \(v_2 \geq (2\text{cost})^{\frac{1}{2}}\).
2.5 References


3. Tasks, Coalition and Reliability

Outline (1)
How can tasks or coalitions between agents be established to improve global profit?
- We define the task allocation problem in precise terms and present different types of contracts between agents. We show that no IR-contract leads to the global optimum, even if all types are allowed.
- Multiagent systems need to be reliable. How can we ensure that? We use a replication approach and probability theory to determine which agents to replicate.
We then consider abstract coalition formation for characteristic function games (CFG), and algorithms for searching the coalition structure graph, and how to distribute the profit among the agents: core of a CFG and Shapley value.

3.1 General Contract Nets

How to distribute tasks?

- Global Market Mechanisms. Implementations use a single centralised mediator.
- Announce, bid, award -cycle. Distributed Negotiation.

We need the following:

1. Define a task allocation problem in precise terms.
2. Define a formal model for making bidding and awarding decisions.

Definition 3.1 (Task-Allocation Problem)

A task allocation problem is given by

1. a set of tasks $T$,
2. a set of agents $A$,
3. a cost function $\text{cost}_i : 2^T \rightarrow \mathbb{R} \cup \{\infty\}$ (stating the costs that agent $i$ incurs by handling some tasks), and
4. the initial allocation of tasks

$$\langle T^{\text{init}}_1, \ldots, T^{\text{init}}_{|A|} \rangle,$$

where $T = \bigcup_{i \in A} T^{\text{init}}_i$, $T^{\text{init}}_i \cap T^{\text{init}}_j = \emptyset$ for $i \neq j$. 
Definition 3.2 (Accepting Contracts, Allocating Tasks)

A contractee $q$ accepts a contract if it gets paid more than the marginal cost of handling the tasks of the contract

$$MC_{\text{add}}(T_{\text{contract}} | T_q) = \text{cost}_q(T_{\text{contract}} \cup T_q) - \text{cost}_q(T_q).$$

A contractor $r$ is willing to allocate the tasks $T_{\text{contract}}$ from its current task set $T_r$ to a contractee, if it has to pay less than it saves by handling them itself:

$$MC_{\text{remove}}(T_{\text{contract}} | T_r) = \text{cost}_r(T_r) - \text{cost}_r(T_r - T_{\text{contract}}).$$

Definition 3.3 (The Protocol)

Agents suggest contracts to others and make their decisions according to the above $MC_{\text{add}}$ and $MC_{\text{remove}}$ sets.

Agents can be both contractors and contractees. Tasks can be recontracted.

- The protocol is domain independent.
- Can only improve at each step: Hill-climbing in the space of all task allocations. Maximum is social welfare: $-\sum_{i \in A} \text{cost}_i(T_i)$.
- Anytime algorithm!

Definition 3.4 (O-, C-, S-, M- Contracts)

A contract is called of type

- **O (Original):** only one task is moved: $\langle T_{i,j}, \rho_{i,j} \rangle$, $|T_{i,j}| = 1$.
- **C (Cluster):** a set of tasks is moved: $\langle T_{i,j}, \rho_{i,j} \rangle$, $|T_{i,j}| \geq 1$.
- **S (Swap):** if a pair of agents swaps a pair of tasks: $\langle T_{i,j}, T_{j,i}, \rho_{i,j}, \rho_{j,i} \rangle$, $|T_{i,j}| = |T_{j,i}| = 1$.
- **M (Multi):** if more than two agents are involved in an atomic exchange of tasks: $\langle T, \rho \rangle$, both are $|A| \times |A|$ matrices. At least 3 elements are non-empty, $|T_{i,j}| \leq 1$.

Figure 15: TSP as Task Allocation Problem.
**Lemma 3.5 (O-Path reaches Global Optimum)**

A path of O-contracts always exists from any task allocation to the optimal one. The length of the shortest such path is at most \(|T|\).

**Definition 3.6 (Task Allocation Graph)**

The task allocation graph has as vertices all possible task allocations (i.e. \(|A||T|\)) and directed edges from one vertex to another if there is a possible contract leading from one to the other.

For O-contracts, searching the graph using breadth-first search takes how much time?

---

We show that it takes time at most \(O(vv^{\frac{1}{2}})\) where \(v = |A||T|\) is the number of vertices.

- Note that from one vertex there are
  \[t_1(|A| - 1) + \ldots + t_n(|A| - 1) = |T|(|A| - 1)\]
  many vertices reachable (O contracts!).

- And breadth first search runs in time \(O(v + e)\), where \(e\) is the number of edges (use an adjacency list, which takes \(O(v^2)\) time to compute).

- Thus \(e = \frac{v}{2}|T|(|A| - 1)\). Thus running time is
  \[O(v + \frac{v}{2}|T|(|A| - 1)) \leq O(|T||A||T| + 1) \leq O(vv^{\frac{1}{2}})\]

**Lemma 3.7 (Allocation graph is sparse)**

We assume that there are at least 2 agents and 2 tasks. We consider the task allocation graph for O contracts. Then the fraction

\[
\frac{\text{number of edges}}{\text{number of edges in the fully connected graph}}
\]

converges to 0 both for \(|T| \to \infty\) as well as \(|A| \to \infty\).
Lemma 3.8 (No IR-Path to Global Optimum)

There are instances where no path of IR O contracts exists from the initial allocation to the optimal one. The length of the shortest IR path (if it exists) may be greater than \(|T|\). But the shortest IR path is never greater than \(|A||T| - (|A| - 1)|T|\).

**Problem:** Local maxima.

A contract may be individually rational but the task allocation is not globally optimal.

Lemma 3.9 (No Path)

There are instances where no path of C-contracts (IR or not) exists from the initial allocation to the optimal one.

**Proof.**

For Lemma 3.9: Two agents and one task. The wrong agent has the task. But no C contract is possible.

Lemma 3.10 (No Path)

There are instances where no path of S-contracts (IR or not) exists from the initial allocation to the optimal one.

Lemma 3.11 (No Path)

There are instances where no path of M-contracts (IR or not) exists from the initial allocation to the optimal one.

$k$-optimal

A task allocation is called $k$-optimal if no beneficial C-contract with clusters of $k$ tasks can be made between any two agents.

Let $m \leq n$. Does

- $m$ optimality imply $n$ optimality;
- $n$ optimality imply $m$ optimality?
**Proof of Lemma 3.10.**

S contracts preserve the number of tasks of an agent. So the optimal allocation can not be reached if the number of tasks are different.

**Proof of Lemma 3.11.**

Suppose there are just two agents and one task. No M task is possible.

**Lemma 3.12 (Reachable allocations for S contracts)**

We assume that there are at least 2 agents and 2 tasks. We consider the task allocation graph for S contracts. Given any vertex \( v \) the fraction

\[
\frac{\text{number of vertices reachable from } v}{\text{number of all vertices}}
\]

converges to 0 both for \(|T| \to \infty\) as well as for \(|A| \to \infty\).

**Proof.**

S contracts preserve the number of tasks of each agent. So any vertex has certain allocations \( t_1, \ldots, t_{|A|} \) for the agents. How many allocations determined by this sequence are there? There are exactly \(|T|!\) many. This is to be divided by \(|A|!|T|!\).

**Theorem 3.13 (All Types necessary)**

For each of the 4 types there exist task allocations where no IR contract with the remaining 3 types is possible, but an IR contract with the fourth type is.
3 Tasks, Coalition and Reliability

3.1 General Contract Nets

Proof.
Consider O contracts (as fourth type). One task and 2 agents: $T_1 = \{t_1\}$, $T_2 = \emptyset$.

$c_1(\emptyset) = 0, c_1(\{t_1\}) = 2, c_2(\emptyset) = 0, c_2(\{t_1\}) = 1$. The O contract of moving $t_1$ would decrease global cost by 1. No C-, S-, or M-contract is possible.

To show the same for C or S contracts, two agents and two tasks suffice.

For M-contracts 3 agents and 3 tasks are needed. 

Theorem 3.14 (O-, C-, S-, M- $\not\implies$ Global Optima)

There are instances of the task allocation problem where no IR sequence from the initial task allocation to the optimal one exists using O-, C-, S-, and M-contracts.

Proof.
Construct cost functions such that the deal where agent 1 gives one task to agent 2 and agent 2 gives 2 tasks to agent 1 is the only one increasing welfare. This deal is not possible with O-, C-, S-, or M-contracts.

Corollary 3.15 (O-, C-, S-, M- $\not\implies$ Global Optima)

There are instances of the task allocation problem where no IR sequence from the initial task allocation to the optimal one exists using any pair or triple of O-, C-, S-, or M-contracts.

Definition 3.16 (OCSM Nets)

A OCSM-contract is a pair $(T, \rho)$ of $|A| \times |A|$ matrices. An element $T_{i,j}$ stands for the set of tasks that agent $i$ gives to agent $j$. $\rho_{i,j}$ is the amount that $i$ pays to $j$.

- How many OCSM contracts are there?
- How much space is needed to represent one?
The number of contracts also depends on \( \rho \) and the values allowed there. However it does not make sense to count these. What we really want is to count the number of matrices \( T \) because for each of them we have to determine whether the contract is beneficial or not. This number is \( v^2 - \frac{v^2 - n}{2} \): roughly \( |A| \times |T| \): quite a lot.

Any OCSM contract can be represented in \( O(|A|^2 + |T|) \) space.

**Theorem 3.17 (OCSM-Nets Suffice)**

Let \( |A| \) and \( |T| \) be finite. If a protocol allows OCSM-contracts, any hill-climbing algorithm finds the globally optimal task allocation in a finite number of steps without backtracking.

**Proof.**

An OCSM contract can move from any task allocation to any other (in one step). So moving to the optimum is IR. Any hill-climbing algorithm strictly improves welfare. As there are only finitely many allocations, the theorem follows. \( \square \)

**Theorem 3.18 (OCSM-Nets are Necessary)**

If a protocol does not allow a certain OCSM contract, then there are instances of the task allocation problem where no IR-sequence exists from the initial allocation to the optimal one.
3 Tasks, Coalition and Reliability
3.1 General Contract Nets

Proof.
If one OCSM contract is not allowed, then the task allocation graph contains two vertices without an edge. We let the initial and the optimal allocation be these two vertices. We construct it in such a way, that all adjacent vertices to the vertex with the initial allocation have lower social welfare. So there is no way out of the initial allocation.

[Andersson98] consider the multiagent version of the TSP problem and apply different sorts of contracts to it.

Several salesmen visit several cities on the unit square. Each city must be visited by exactly one salesman. They all have to return home and want to minimise their travel costs.

salesman = agent, task = city

Experiments with up to 8 agents and 8 tasks. Initial allocation randomly chosen.

Ratio bound (welfare of obtained local optimum divided by global optimum) and mean ratio bound (over 1000 TSP instances) were computed (all for fixed number of agents and tasks). Global optimum was computed using IDA*.

A protocol consisting of 5 intervals is considered. In each interval, a particular contract type was considered (and all possible contracts with that type): 1024 different sequences.

In each interval a particular order for the agents and the tasks is used. First all contracts involving agent 1. Then all involving agent 2 etc. Thus it makes sense to have subsequent intervals with the same contract type.

<table>
<thead>
<tr>
<th>Order</th>
<th>No. Sequence</th>
<th>Social Welfare</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>OCOCO</td>
<td>1.03113</td>
</tr>
<tr>
<td>2</td>
<td>OOCCO</td>
<td>1.03268</td>
</tr>
<tr>
<td>3</td>
<td>OOCOC</td>
<td>1.03276</td>
</tr>
<tr>
<td>4</td>
<td>OOCOC</td>
<td>1.03279</td>
</tr>
<tr>
<td>5</td>
<td>OCOCO</td>
<td>1.03413</td>
</tr>
<tr>
<td>6</td>
<td>SOCCO</td>
<td>1.03488</td>
</tr>
<tr>
<td>7</td>
<td>SSOCC</td>
<td>1.03536</td>
</tr>
<tr>
<td>8</td>
<td>COCCO</td>
<td>1.03555</td>
</tr>
<tr>
<td>9</td>
<td>OCOCO</td>
<td>1.03857</td>
</tr>
<tr>
<td>10</td>
<td>MCCOC</td>
<td>1.03945</td>
</tr>
<tr>
<td>11</td>
<td>OCOCC</td>
<td>1.03954</td>
</tr>
<tr>
<td>12</td>
<td>MOCOC</td>
<td>1.03988</td>
</tr>
<tr>
<td>13</td>
<td>MCOCO</td>
<td>1.04001</td>
</tr>
<tr>
<td>14</td>
<td>MCCOC</td>
<td>1.04304</td>
</tr>
<tr>
<td>15</td>
<td>COCCO</td>
<td>1.04407</td>
</tr>
</tbody>
</table>

Table 1: The best Contract sequences.
3 Tasks, Coalition and Reliability

3.1 General Contract Nets

Table 2: Best, average and worst Contracts.

<table>
<thead>
<tr>
<th>Order No.</th>
<th>Sequence</th>
<th>Social Welfare</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>OCOCO</td>
<td>1.03113</td>
</tr>
<tr>
<td>2</td>
<td>OOCO</td>
<td>1.02588</td>
</tr>
<tr>
<td>3</td>
<td>OCCOC</td>
<td>1.0276</td>
</tr>
<tr>
<td>4</td>
<td>OOOC</td>
<td>1.02297</td>
</tr>
<tr>
<td>5</td>
<td>C-local</td>
<td>1.13557</td>
</tr>
<tr>
<td>565</td>
<td>O-local</td>
<td>1.2025</td>
</tr>
<tr>
<td>579</td>
<td>OOOOO</td>
<td>1.21298</td>
</tr>
<tr>
<td>696</td>
<td>CCCCC</td>
<td>1.23515</td>
</tr>
<tr>
<td>1021</td>
<td>CSSSS</td>
<td>1.61181</td>
</tr>
<tr>
<td>1022</td>
<td>CMMMM</td>
<td>1.65965</td>
</tr>
<tr>
<td>1023</td>
<td>MMMMM</td>
<td>1.76634</td>
</tr>
<tr>
<td>1024</td>
<td>SSSSS</td>
<td>1.89321</td>
</tr>
</tbody>
</table>

Figure 16: Ratio bounds (second graph is zoomed in).

Figure 17: Ratio bounds for the 4 best and single type sequences.

Figure 18: Contracts performed and tried.
3.2 Coalition Formation in CFG’s

Definition 3.19 (Characteristic Function Game (CFG))

A characteristic function game is a tuple \((A, v)\) where \(A\) is a finite set (of agents) and \(v : 2^A \rightarrow \mathbb{R}^+_0; S \mapsto v(S)\).
We assume \(v(\emptyset) = 0\) and call \(v(S)\) the value of coalition \(S\).

Thus the value is independent of the nonmembers. But

1. Positive Externalities: Overlapping goals. Nonmembers perform actions and move the world closer to the coalition’s goal state.
2. Negative Externalities: Shared resources. Nonmembers may use up the resources.

Definition 3.20 (Coalition Structure CS)

A coalition structure \(CS\) over the set \(A\) is any partition \(\{C^1, \ldots, C^k\}\) of \(A\), i.e. \(\bigcup_{j=1}^k C^j = A\) and \(C^i \cap C^j = \emptyset\) for \(i \neq j\). We denote by \(CS_M\) the set of all coalition structures \(CS\) over the set \(M \subseteq A\).

Finally, we define the social welfare of a coalition structure \(CS\) by

\[ v(CS) := \sum_{C \in CS} v(C). \]
Definition 3.21 (Coalition Formation in CFG’s)

Coalition Formation in CFG’s consists of:

**Forming CS**: formation of coalitions such that within each coalition agents coordinate their activities.

**Solving Optimisation Problem**: For each coalition in a CS, the tasks and resources of the agents have to be pooled. **Maximise monetary value**.

**Payoff Division**: Divide the value of the generated solution among agents.

Maximise the social welfare of the agents $A$ by finding a coalition structure

$$CS^* = \arg \max_{CS \in CS} v(CS),$$

where

$$v(CS) := \sum_{S \subseteq CS} v(S).$$

How many coalition structures are there?

Let $Z(|A|, i)$ denote the number of coalition structures with $i$ coalitions. Then

- $Z(|A|, |A|) = Z(|A|, 1) = 1$.
- $Z(|A|, i) = i Z(|A| - 1, i) + Z(|A| - 1, i - 1)$.
  - Add one agent to a game with $|A| - 1$ agents.
- $\sum_{i=1}^{A} Z(|A|, i)$ is the number of coalition structures.

In total: number of coalitions is bounded below by $(|A| \choose 2)$.

**Figure 19**: Number of Coalition (Structures).
Approximation of $v(CS)$.

How can we approximate $v(CS)$?

Choose set $N \subseteq CS_A$ and pick the best coalition seen so far:

$$CS_N^* = \arg \max_{CS \in N} v(CS).$$

We want our approximation as good as possible.

We want to find a small $k$ and a small $N$ such that

$$\frac{v(CS^*)}{v(CS_N^*)} \leq k.$$

$k$ is the bound (best value would be 1) and $N$ is the part of the graph that we have to search exhaustively.

We consider 3 search algorithms:

- **MERGE**: Breadth-first search from the top.
- **SPLIT**: Breadth first from the bottom.
- **Coalition-Structure-Search (CSS1)**: First the bottom 2 levels are searched, then a breadth-first search from the top.

**MERGE** might not even get a bound, without looking at all coalitions.

**SPLIT** gets a good bound ($k = |A|$) after searching the bottom 2 levels (see below). But then it can get slow.

**CSS1** combines the good features of **MERGE** and **SPLIT**.
Why is SPLIT slow after the first two bottom levels?
Construct a bad example as follows.

\[ v(S) = \begin{cases} 
1, & \text{if } |S| = 1; \\
0, & \text{otherwise.}
\end{cases} \]

So the optimum is the top node, and

\[ v(C^*) = \frac{a}{l - 1}, \]

where \( l \) is the level that the algorithm has completed (the number of unit coalitions on a level \( l \) is always \( \leq l - 1 \) except the top level where it is equal to \( l \), namely \(|A|\)).

---

**Theorem 3.22 (Minimal Search to get a bound)**

To bound \( k \), it suffices to search the lowest two levels of the CS-graph. Using this search, the bound \( k = |A| \) can be taken. This bound is tight and the number of nodes searched is \( 2|A| - 1 \).

No other search algorithm can establish the bound \( k \) while searching through less than \( 2|A| - 1 \) nodes.

---

**Proof.**

There are at most \(|A|\) coalitions included in \( C^* \). Thus

\[ v(C^*) \leq |A| \max_S v(S) \leq |A| \max_{CS \in N} v(CS) = |A| v(C^*_N) \]

Number of coalitions at the second lowest level: \( 2^{|A|} - 2 \).

Number of coalition structures at the second lowest level:

\[ \frac{1}{2} (2^{|A|} - 2) = 2^{A-1} - 1. \]

Thus the number of nodes visited is: \( 2^{A-1} \).

---

What exactly does the last theorem mean? Let \( n_{\text{min}} \) be the smallest size of \( N \) such that a bound \( k \) can be established.

**Positive result:** \( \frac{n_{\text{min}}}{\text{partitions of } A} \) approaches 0 for \(|A| \to \infty\).

**Negative result:** To determine a bound \( k \), one needs to search through exponentially many coalition structures.
Algorithm (CS-Search-1)

The algorithm comes in 3 steps:

1. Search the bottom two levels of the CS-graph.
2. Do a breadth-first search from the top of the graph.
3. Return the CS with the highest value.

This is an anytime algorithm.

Experiments

6-10 agents, values were assigned to each coalition using the following alternatives:

1. values were uniformly distributed between 0 and 1;
2. values were uniformly distributed between 0 and \(|\mathcal{A}|\);
3. values were superadditive;
4. values were subadditive.

Theorem 3.23 (CS-Search-1 up to Layer l)

With the algorithm CS-Search-1 we get the following bound for \(k\) after searching through layer \(l\):

\[
\begin{cases}
\left\lfloor \frac{|\mathcal{A}|}{h} \right\rfloor & \text{if } |\mathcal{A}| \equiv h - 1 \mod h \text{ and } |\mathcal{A}| \equiv l \mod 2, \\
\left\lceil \frac{|\mathcal{A}|}{h} \right\rceil & \text{otherwise.}
\end{cases}
\]

where \(h = \lfloor \frac{|\mathcal{A}| - l}{2} \rfloor + 2\).

Thus, for \(l = |\mathcal{A}|\) (check the top node), \(k\) switches from \(|\mathcal{A}|\) to \(\frac{|\mathcal{A}|}{2}\).
3 Tasks, Coalition and Reliability

3.2 Coalition Formation in CFG's

Figure 22: SPLIT.

Figure 23: CS-Search-1.

Figure 24: Subadditive Values.

Figure 25: Superadditive Values.
3 Tasks, Coalition and Reliability

3.2 Coalition Formation in CFG’s

Figure 26: Coalition values chosen uniformly from [0, 1].

Figure 27: Coalition values chosen uniformly from [0, |S|].

Figure 28: Comparing CS-Search-1 with SPLIT.

1. Is CS-Search-1 the best anytime algorithm?
2. The search for best \( k \) for \( n' > n \) is perhaps not the same search to get best \( k \) for \( n \).
3. CS-Search-1 does not use any information while searching. Perhaps \( k \) can be made smaller by not only considering \( v(CS) \) but also \( v(S) \) in the searched \( CS' \).
3.3 CFG’s and the Core

Idea: Consider a protocol (to build coalitions) as a game and consider Nash-equilibrium.

Problem: Nash-Eq is too weak!

Definition 3.24 (Strong Nash Equilibrium)
A profile is in strong Nash-Eq if there is no subgroup that can deviate by changing strategies jointly in a manner that increases the payoff of all its members, given that nonmembers stick to their original choice.

This is often too strong and does not exist.

Definition 3.25 (Monotone Games)
A CFG \((A, v)\) is called monotone, if
\[ v(C) \leq v(D), \]
for every pair of coalitions \(C, D \subseteq A\) such that \(C \subseteq D\).

Many games have this property, but there may be communication/coordination costs. Or some players hate others and do not want to be in the same coalition. The next slide introduces a strictly stronger condition.

Definition 3.26 (Superadditive Games)
A CFG \((A, v)\) is called superadditive, if
\[ v(S \cup T) \geq v(S) + v(T), \]
where \(S, T \subseteq A\) and \(S \cap T = \emptyset\).

Lemma 3.27
Coalition formation for superadditive games is trivial.

Conjecture
All games are superadditive.
The conjecture is wrong, because the coalition process is not for free: communication costs, penalties, time limits.

**Definition 3.28 (Subadditive Games)**

A CFG \((A, v)\) is called **subadditive**, if

\[ v(S \cup T) \leq v(S) + v(T), \]

where \(S, T \subseteq A\) and \(S \cap T = \emptyset\).

Coalition formation for subadditive games is trivial.

**Superadditive Cover**

**Definition 3.29 (Superadditive Cover)**

Given a game \(G = (A, v)\) that is not superadditive, we can transform it to a superadditive game \(G^* = (A, v^*)\) as follows

\[ v^*(C) := \max_{CS \in C} C v(C) \]

This game is called the **superadditive cover** of \(G\).

**Convex Games**

**Definition 3.30 (Convex Game)**

A CFG \((A, v)\) is **convex**, if for all coalitions \(T, S\) with \(T \subseteq S\) and each player \(i \in A \setminus S\):

\[ v(S \cup \{i\}) - v(S) \geq v(T \cup \{i\}) - v(T) \]

Convex games are superadditive. Superadditive games are monotone. The other directions do not hold.
Sierra Madre Game: $v(S) = \lfloor \frac{|S|}{2} \rfloor$.

Majority Game: $v(S) = \begin{cases} 1, & \text{if } |S| = 1; \\ \alpha, & \text{if } |S| = 2; \\ 0, & \text{if } |S| = 1. \end{cases}$

**Example 3.33 (Parliament)**

Suppose there are four parties and the result of the elections is as follows:

1. A: 45 %,
2. B: 25 %,
3. C: 15 %,
4. D: 15 %.

Wlog we assume the overall payoff (grand coalition) is 1.

**Definition 3.34 (Payoff Vector)**

A payoff vector for a CFG and a coalition structure $CS$ is a tuple $(x_1, \ldots, x_n)$ such that

1. $x_i \geq 0$ and $\sum_{i=1}^{n} x_i = v(A)$ ($n = |A|$),
2. $\forall C \in CS : \sum_{i \in C} x_i \geq v(C)$.

Note that the last condition is only supposed to hold for all coalitions in the given coalition structure.

**Definition 3.35 (Core of a CFG)**

The core of a CFG is the set of all pairs $(CS, (x_1, \ldots, x_n))$ of coalition structures ($CS \in CS_A$) and payoff vectors such that the following holds:

$\forall S \subseteq A : \sum_{i \in S} x_i \geq v(S)$

Here, the condition is supposed to hold for all $S$. We do not want any set of agents to form a new coalition.
When the grand coalition forms, we can simplify the last definition.

**Definition 3.36 (Core of Superadditive Games)**

The core of a superadditive CFG is the set of all payoff vectors such that the following holds:

\[
\forall S \subseteq A : \sum_{i \in S} x_i \geq v(S)
\]

Thus the core corresponds to the strong Nash equilibrium mentioned in the beginning.

What about the core in the above two examples?

**Sierra Madre:**
- **Case 1:** \(|A| \geq 4\) and \(|A|\) is even. Then the core consists of a single payoff vector \(\langle \frac{1}{2}, \ldots, \frac{1}{2} \rangle\).
- **Case 2:** \(|A| \geq 3\) and \(|A|\) is odd. Then the core is empty.

**3 player majority game:** The core consists of all payoff vectors that assign 1 to the grand coalition and something \(\geq \alpha\) to all coalitions with two agents. Thus we have three cases:
- **Case 1:** \(\alpha \leq \frac{2}{3}\). Then the core consists of the payoff vectors \(\langle \frac{1}{2}, \frac{1}{2}, 1 - \alpha \rangle, \langle \frac{1}{2}, 1 - \alpha, \frac{1}{2} \rangle, \langle 1 - \alpha, \frac{1}{2}, \frac{1}{2} \rangle\).
- **Case 2:** \(\alpha = \frac{2}{3}\). Then the core consists of the vector \(\langle \frac{1}{3}, \frac{1}{3}, \frac{1}{3} \rangle\).
- **Case 3:** \(\alpha \geq \frac{2}{3}\). Then the core is empty.

**Lemma 3.37**

If \(\langle CS, \langle x_1, \ldots, x_n \rangle \rangle\) is in the core of a CFG \(\langle A, v \rangle\), then \(v(CS) \geq v(CS')\) for all coalition structures \(CS' \in CS_A\).

**Proof.**

We can write \(v(CS) = \sum_{i \in A} x_i = \sum_{C' \in CS'} x(C')\) and \(v(CS') = \sum_{C' \in CS'} v(C')\). Because of the definition of the core, \(x(C') \geq v(C')\) for all \(C'\) and therefore \(v(CS) \geq v(C')\).
Theorem 3.38

Let a CFG $G = (A, v)$ be given (not necessarily superadditive). Then $G$ has a non-empty core if and only if its superadditive cover $G^*$ has a non-empty core.

Proof \rightarrow exercise

Theorem 3.39

Each convex CFG $G = (A, v)$ has a non-empty core.

Proof.

Let $\pi$ be a permutation of $A$ and let $S_\pi(i)$ be the set of all predecessors of $i$ wrt. $\pi$. We claim that for $x_i := v(S_\pi(i) \cup \{i\}) - v(S_\pi(i))$, the core of $G$ contains $\langle x_1, \ldots, x_n \rangle$. It is easy to show that all $x_i$ are greater or equal to 0 and that they all sum up to the value of the game ($\rightarrow$ exercise).

(Proof of Theorem 3.39, cont.)

Assume there is a coalition $C = \{i_1, \ldots, i_s\}$ such that $v(C) \geq x(C)$. Wlog we assume $\pi(i_1) \leq \ldots \leq \pi(i_s)$. Obviously

$v(C) = v(\{i_1\}) - v(\emptyset) + v(\{i_1, i_2\}) - v(\{i_1\}) + \ldots + v(C)v(C \setminus \{i_s\})$

Because of convexity (let $T_j := \{i_1, \ldots, i_{j-1}\}, S_j := \{1, 2, \ldots, i_j - 1\}$ and apply the definition of convexity) for all $j$:

$v(T_j \cup \{i_j\}) - v(T_j) \leq v(S_j \cup \{i_j\}) - v(S_j) = x_i$

Adding these pairs up, we get $v(C) \leq x(C)$, which is a contradiction.
The payoff division should be fair between the agents, otherwise they leave the coalition.

**Definition 3.40 (Dummies, Interchangeable)**

Agent \( i \) is called a **dummy**, if for all coalitions \( S \) with \( i \not\in S \):
\[
u(S \cup \{i\}) - \nu(S) = \nu(\{i\}).\]

Agents \( i \) and \( j \) are called **interchangeable**, if for all coalitions \( S \) with \( i \in S \) and \( j \not\in S \):
\[
u(S \setminus \{i\} \cup \{j\}) = \nu(S).

**Axioms for Payoff Division**

- **Efficiency**: 
  \[
  \sum_{i \in A} x_i = \nu(A).
  \]

- **Symmetry**: If \( i \) and \( j \) are interchangeable, then \( x_i = x_j \).

- **Dummies**: For all dummies \( i \):
  \[
  x_i = \nu(\{i\}).
  \]

- **Additivity**: For any two games \( v, w \):
  \[
  x_{v \oplus w} = x_v + x_w,
  \]
  where \( v \oplus w \) denotes the game defined by
  \[
  (v \oplus w)_S = \nu(S + w(S).
  \]

**Theorem 3.41 (Shapley-Value)**

For a CFG \( G = (A, \nu) \) there is only one payoff division satisfying the above four axioms. It is called the **Shapley value** of agent \( i \) and is defined by
\[
\phi_i(G) = \frac{1}{|A|!} \sum_{S \subseteq A} (|A| - |S| - 1)! |S|!(\nu(S \cup \{i\}) - \nu(S))
\]
View coalition formation by adding one agent at a time. For a given sequence (of how to add agents), what is agent $i$’s marginal contribution (at the time it is added to the set $S$)?

It is $\left( v(S \cup \{i\}) - v(S) \right)$.

- How many ways to form $S$ (before $i$ joined): $|S|!$.
- How many ways to form $S$ (after $i$ joined): $\left(|A| - |S| - 1\right)!$.
- This has to be summed up over all possible sets $S$ and averaged over all $|A|!$ orderings of the agents.

### 3.4 Payoff Division

**Definition 3.42 (Banzhaf Index)**

For a CFG $G = (A, v)$ the Banzhaf Index of agent $i$ is

$$
\beta_i(G) = \frac{1}{2^{|A|-1}} \sum_{C \subseteq A \setminus \{i\}} (v(C \cup \{i\}) - v(C))
$$

The Banzhaf Index satisfies all axioms but **efficiency**.

The following normalised Banzhaf index $\eta_i(G)$ satisfies efficiency, but fails in additivity ($\Rightarrow$ exercise):

$$
\eta_i(G) := \frac{\beta_i(G)}{\beta_i(G)} v(A).
$$

### 3.5 Reliability

We are showing one of the approaches that build more reliable multiagent systems.

Hopefully, you will get some idea of how we usually approach interesting research problems.
Reliability of multiagent systems has become one of the critical issues for commercial and industrial applications. It must be addressed to achieve wider adoption: [Luck et al. 2006]

Why?
In real-world environments, external events can easily cause a MAS to crash.

Some existing approaches:
(1) Extensive reliability testing: the environment is simulated and a large variety of scenarios are played out;
(2) MAS debugging tools: information gathering and visualisation; offline debugging; online monitoring of agents’ behaviours;
(3) Agent replication: creates one or more replicas for one or more agents in a multiagent system. Each of the replicas is able to perform the same task as the original agent.

We will introduce a replication based approach.

Definition 3.43 (MAS Application, Network)

1. Agents $a, b, \ldots$: provide one or more services; located on a host computer; require resources.
2. Multiagent Application $\mathcal{M}$: consist of a finite set of agents; agents cooperate to provide services; locate on a network of hosting nodes.
3. A Network $\mathcal{N} = (V, E)$:
   - $V$ is the set of nodes in the network;
   - $E = V \times V$: fully connected overlay network;
   - for each $n \in V$, $\text{space}(n)$ denotes the fixed amount of memory.
   - disconnect probability of nodes $dp$: a disconnect probability function for a network is a mapping $dp : \mathcal{N} \rightarrow \mathbb{C}[0, 1]$

Example 3.44 (Disconnect Probability)

$dp(n_1) = [0.2]$ says that there is a 20% probability that the node $n_1$ will get disconnected.
Definition 3.45 (Deployment $\mu$)

Given a network $\mathcal{N} = (V, E)$ and a MAS $\mathcal{M} = \{a_1, \ldots, a_n\}$, a deployment is a mapping $\mu : V \to 2^\mathcal{M}$ such that for all $1 \leq i \leq n$, there exists a $n \in V$ such that $a_i \in \mu(n)$.

A deployment $\mu$ must satisfy the following constraints:

1. **deployment constraint:** each agent must be deployed somewhere.
2. **resource constraint:** the agents deployed at a node cannot use more memory than that node makes available.

Example 3.46 (Deployment)

Suppose a network $\mathcal{N} = \{V, E\}$ where $V = \{n_1, n_2, n_3, n_4\}$, a multiagent application $\mathcal{M} = \{a, b, c, d\}$, and a deployment $\mu$ w.r.t. $\mathcal{M}, \mathcal{N}$ are given as follows:

- $\mu(n_1) = \{a\}$,
- $\mu(n_2) = \{b, d\}$,
- $\mu(n_3) = \{a, b\}$,
- $\mu(n_4) = \{b, c\}$.

Definition 3.47 (Survivability of a Deployment)

The survival of a multiagent system means that at any time, at least one replica of each agent in the system must be available on at least one node in the network.

Survivability of a deployment is the probability with which the multiagent system will survive.

Three Questions:

- **Q1:** Given a deployment $\mu$, how to compute the survival probability of $\mu$?
- **Q2:** Given a MAS $\mathcal{M}$ and a network $\mathcal{N}$, how to find a best deployment which has the maximal survivability?
- **Q3:** Given the current deployment $\mu_{old}$ within the dynamic environment, how to make sure $\mu$ can survive dynamically and adaptively to the changing environment?
Definition 3.48 (Survivability Function \( SF \))

A survivability function \( SF : (\mu, dp) \rightarrow [0, 1] \) is any function which takes a deployment \( \mu \) and a disconnect probability function \( dp \) as input, and returns the probability with which it is guaranteed that the MAS will survive.

Assumption on Dependencies of \( dp \)'s

The disconnect probabilities of nodes could be independent, dependent or ignorant. Here, we assume the node failures are independent of one another.

Node-based algorithm \( SF_n \)

The survivability of \( \mu \) is the probability that one of the valid future networks \( N_i \) will survive!

Suppose \( N_i = \{n_{i1}, n_{i2}, \cdots n_{il}\} \), where \( n_{ij} \in V \) is a valid future network. The survivability of \( N_i \) is given by:

\[
surv(N_i) = \prod_{n_p \in N_i} (1 - dp(n_p)) \cdot \prod_{n_q \in V \setminus N_i} dp(n_q)
\]

(1)

Definition 3.49 (Possible Future Network of \( N \))

A possible future network of \( N(V, E) \) consists of a subset of the nodes and a subset of the edges involving the selected nodes.

Definition 3.50 (Valid Future Network)

Given a deployment \( \mu \), suppose \( N_i \) is a possible future network and \( \mu \) is a valid deployment w.r.t. \( N_i \). Then we say \( N_i \) is a valid possible future network.

We say that \( \mu \) is valid w.r.t. \( N_i \) if and only if for each agent \( a \in M \), \( \{n \mid a \in \mu(n)\} \cap N_i \neq \emptyset \).

Node-based algorithm \( SF_n \)

Let \( N \) be the set of all valid networks. Then the survivability of \( \mu \) is the sum of the survivability of each of valid future networks \( N_i \)’s.

Survivability of \( \mu \)

\[
SF_n(\mu) = \sum_{N_i \in N} surv(N_i) = surv(N_1 \cup N_2 \cup \ldots \cup N_{|N|})
\]
Example 3.51

Consider a network with $V = \{n_1, n_2, n_3\}$ and a multiagent application $M = \{a_1, a_2, a_3\}$. The disconnect probability $dp$ is: $dp(n_1) = 0.7$, $dp(n_2) = 0.6$, $dp(n_3) = 0.4$. Suppose the deployment $\mu$ is given by:

$\mu(n_1) = \{a_1, a_2, a_3\}$, $\mu(n_2) = \{a_1\}$, $\mu(n_3) = \{a_2, a_3\}$.

Thus, the valid future networks are: $N_1 = \{n_1\}$, $N_2 = \{n_2, n_3\}$, $N_3 = \{n_1, n_2, n_3\}$, $N_4 = \{n_1, n_2\}$, $N_5 = \{n_1, n_3\}$.

The survivability of each network is:

$\text{surv}(N_1) = (0.3)(0.6)(0.4) = 0.072$, and

$\text{surv}(N_2) = (1 - 0.6)(1 - 0.4)(0.7) = 0.168$.

Similarly we have $\text{surv}(N_3) = 0.072$, $\text{surv}(N_4) = 0.048$, $\text{surv}(N_5) = 0.108$.

The survivability of the deployment is the sum of the probability that $N_i$ survives:

$SF_n(\mu) = 0.072 + 0.168 + 0.072 + 0.048 + 0.108 = 0.468$.

Proof (1):

The proof of NP-hardness uses a reduction from set-covering. Given a finite set $X$ and a family $\mathcal{F}$ of subsets of $X$, such that each element of $X$ belongs to at least one subset in $\mathcal{F}$: $X = \bigcup_{S \in \mathcal{F}} S$. A subset $S \in \mathcal{F}$ covers its elements. The decision version of the set-covering problem asks whether or not a covering $N \subseteq \mathcal{F}$ whose members cover all of $X$ exists with size at most $k$. The set-covering problem is NP-complete: [CLRS01].

Note.

We have to find all subsets of $V$ w.r.t. which $\mu$ is a valid deployment. $SF_1_n$ is exponential in the number of nodes.

Theorem 3.52 (Complexity)

The problem of computing the survivability of a given deployment under the independence assumption is at least NP-hard.

Any algorithm to compute survivability is exponential, unless the PH collapses.

Proof (2):

The figure below illustrates how set-covering is transformed into the problem of finding valid networks. Clearly, $N$ is an existing covering with size $k$ if and only if $N$ is a valid network with number $k$ of nodes w.r.t $\mu$.
Survivability of a given deployment.

Note that we are only interested in heuristics which are:

- fast (polynomial-time) algorithms;
- underestimation of the actual survivability.

Solution!

To develop heuristic algorithms to compute the survivability of a given deployment.

3. Only the $\alpha$ vertices with the highest probability are further expanded in the same way.
4. If a vertex labelled $N_i$ is expanded, its children will be labelled by $N_i \setminus \{n\}$ for each node $n \in N_i$. Again only $\alpha$ vertices will be expanded and so on.
5. We stop when there are no more nodes to expand. $SF3$ sums the probability of all the valid future networks in the search tree.

$SF_n$ is too expensive for real-world applications!

$SF3$, tree-based heuristic

In $SF_n$, we compute and add the probabilities of all valid future networks. $SF3$ attempts to find only a subset of all valid networks.

$SF3$ does this via a tree search in which the root of each node is labelled with a subset of $V$.

1. The root is labelled with $V$. The probability of $V$ is computed.
2. For each node $n \in V$ there is a vertex labelled $V \setminus \{n\}$ in the second level of the tree. For all such valid vertices, the probability of their labelled possible future networks is computed.

Example 3.53 ($SF3$)

Consider a deployment $\mu(n_1) = \{a_1, a_2\}$, $\mu(n_2) = \{a_1\}$, and $\mu(n_3) = \{a_2\}$. Let $\alpha = 1$. The root is $V = \{n_1, n_2, n_3\}$. Thus $X_0 = \{n_1, n_2, n_3\}$, and $\text{surv}(X_0) = (1 - dp(n_1))(1 - dp(n_2))(1 - dp(n_3)) = (1 - 0.7)(1 - 0.6)(1 - 0.4) = 0.072$.

In the next level of the graph, 3 subsets are generated:

- $X_{\alpha 1} = \{n_2, n_3\}$, and $\text{surv}(X_{\alpha 1}) = (1 - 0.6)(1 - 0.4)(0.7) = 0.168$;
- $X_{\alpha 12} = \{n_1, n_3\}$, and $\text{surv}(X_{\alpha 12}) = (1 - 0.7)(1 - 0.4)(0.6) = 0.108$;
- $X_{\alpha 13} = \{n_1, n_2\}$, and $\text{surv}(X_{\alpha 13}) = (1 - 0.7)(1 - 0.6)(0.4) = 0.048$.

As $\alpha = 1$, we use the set $X_{\alpha 11}$ to generate subsets in the next level:

- $X_{\alpha 11} = \{n_2, n_3\}$ is removed because it is not valid;
- $X_{\alpha 12} = \{n_3\}$ is invalid thus removed.

The search terminates because no more valid subsets can be created. The survivability of the deployment is:

$SF3(\mu) = \text{surv}(X_0) + \text{surv}(X_{\alpha 11}) + \text{surv}(X_{\alpha 12}) + \text{surv}(X_{\alpha 13}) = 0.396$. 
### Theorem 3.54

*SF*3 is an underestimate* of *SF*ₙ. Suppose α is fixed, the time complexity to compute *SF*3 is O(α |V|² log(α |V|) + α |V|² |M|), that is, the computation is polynomial if α is a fixed constant.

### However, if α is polynomial in V, SF3 considers only a polynomial number of future networks. Therefore it may return very poor results (in terms of approximation ratio) if there is a large number of nodes.

### Some other heuristic algorithms

- *SF*2: anytime algorithm
- *SF*4: disjoint based algorithm
- *SF*₄₉: group based algorithm
- *SF*5: split based algorithm.

### Evaluations

- **running time**: time taken to compute the given deployment
- **approximation ratio**: survivability returned by heuristics/actual survivability

### Various experimental settings

- problem’s size: number of agents + number of nodes
- number ratio: number of agents/number of nodes
- space ratio: avg. of memory requirement of agents/avg. of memory available on nodes.
- various distributions of nodes’ disconnect probabilities.

### Computation time and survivability returned by *SF*3 with varying α

![Graph showing survivability computed by SF3](image)
Computation time and survivability returned by SF3 with varying $\alpha$.

### Table 3: Approximation ratio (with space ratio (2-3))

<table>
<thead>
<tr>
<th>Problem size</th>
<th>Anytime algo.</th>
<th>SF3</th>
<th>SF4</th>
<th>Split algo.</th>
<th>Group algo.</th>
</tr>
</thead>
<tbody>
<tr>
<td>n18, a30</td>
<td>0.951475</td>
<td>0.998919</td>
<td>0.879377</td>
<td>0.999515</td>
<td>0.948644</td>
</tr>
<tr>
<td>n24, a40</td>
<td>0.85197</td>
<td>0.964037</td>
<td>0.836836</td>
<td>0.965813</td>
<td>0.923682</td>
</tr>
<tr>
<td>n30, a50</td>
<td>0.899324</td>
<td>0.937145</td>
<td>0.775934</td>
<td>0.939865</td>
<td>0.893412</td>
</tr>
</tbody>
</table>

### Table 4: Approximation ratio (with space ratio (3-4))

<table>
<thead>
<tr>
<th>Problem size</th>
<th>Anytime algo.</th>
<th>SF3</th>
<th>SF4</th>
<th>Split algo.</th>
<th>Group algo.</th>
</tr>
</thead>
<tbody>
<tr>
<td>n18, a30</td>
<td>0.981061</td>
<td>0.98647</td>
<td>0.96855</td>
<td>0.998956</td>
<td>0.98707</td>
</tr>
<tr>
<td>n24, a40</td>
<td>0.979052</td>
<td>0.991792</td>
<td>0.978524</td>
<td>0.99866</td>
<td>0.994861</td>
</tr>
<tr>
<td>n30, a50</td>
<td>0.98326</td>
<td>0.975502</td>
<td>0.982723</td>
<td>0.998131</td>
<td>0.994358</td>
</tr>
</tbody>
</table>

SF3 works well with some settings.

Result

Different algorithms work well in different environment settings and applications.

See our paper for details: [Zhang et al. 2005, Zhang 2005].

We introduced how to compute the survivability of a deployment given:

1. a multiagent system $\mathcal{M}$;
2. a network $\mathcal{N} = \{V, E\}$ and its disconnect probability $dp$; and
3. a fixed deployment $\mu$ w.r.t $\mathcal{M}$ and $\mathcal{N}$.

Optimal deployment

Assume that we are given only 1. and 2., and our task is to find an optimal deployment $\mu$ w.r.t $\mathcal{M}$ and $\mathcal{N}$ which has the maximal survivability.

Remember that deployments must satisfy (1) space constraints and (2) deployment constraints.
Branch and Bound algorithm

- **Initial state:** all agents on all nodes; if it is a valid deployment, then stop;
- **Children of a state** are all the states that are obtained by the removal of one agent from one node of the state.
- **In each stage:** if a valid deployment is found, compute its survivability and use to bound the search.

Finding the optimal deployment involves **two sources of complexity:**
- The exponential space of possible deployments, and
- Computing the probability of survival is exponential or polynomial (heuristics).

**Theorem 3.55**
The problem of finding an optimal deployment is at least **NP-hard.**

Even if the fast algorithms are used to compute the deployments, the process of finding the optimal deployment can be very **time-consuming!**

**Question**
Can you come up with any fast heuristics to find sub-optimal deployments?

Refer to [Kraus et al. 2003] for two heuristics: **agent-based** and **node-based** heuristics.
What we have introduced above is **centralised, static survivability**, which means,

the MAS is deployed over a given network of host nodes. A special “survivability algorithm” is placed on a node selected by the MAS developer.

**Problems of centralised approach**

1. The survivability algorithm itself resides on a single node—single point of failure.
2. The environment is dynamic and changing. Thus the survivability algorithms needs to adapt to changes that affect the survivability of the MAS.

**Solution**

Add a special **deployment agent (da)** to one or more nodes in the network. Note we distinguish **da** from regular agents.

If disconnect probabilities of nodes change, **da** will **re-evaluate** the survivability of the deployment (using survivability algorithm), then **re-deploy** the regular agents to the new locations.

**Two issues**

- **Location of da**: where to deploy the deployment agents?
- **Behaviours of da**: how **da** re-deploys regular agents?

**Three distributed algorithms:**

<table>
<thead>
<tr>
<th>location of da</th>
<th>behaviour of da</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASA-1</td>
<td>a copy of <strong>da</strong> to each node</td>
</tr>
<tr>
<td>ASA-2</td>
<td>Some of nodes</td>
</tr>
<tr>
<td>ASA-3</td>
<td>Some of nodes</td>
</tr>
</tbody>
</table>
Key step is (4). During execution of the algorithm,
1. at least one copy of each agent must be on the network at all times;
2. no host’s space should be exceeded.

ASA-2 (algorithm sketch)
- \( \mathcal{M}^* = \mathcal{M} \cup \{\text{da}\} \);
- Once \text{da} is told to redeploy by an external process, apply the deployment algorithm on \( \mathcal{M}^* \). This returns a new deployment \( \mu_{\text{new}} \) w.r.t \( \mathcal{M}^* \) (including \text{da});
- The algorithm deletes all but one copy of all agents.
- It then moves/copies remaining agents to their new locations.

ASA-3
- **Location of da**: similar to ASA-2, where \text{da} is considered as regular agents;
- **Behaviours of da**: similar to ASA-1, where uses difference table and rules based algorithm.

Rules (partial list) for step 4:
- \( \text{DEL}(A, n) \leftarrow (\forall a \in A) \text{safeDel}(a, n) \)
- \( \text{remdif}(A, \text{Remv}, n) \land \text{remdif}(A, \text{Deploy}, n) \leftarrow \text{DEL}(A, n) \)
- \( \text{ADD}(a, n) \leftarrow (\text{space}(n) - \text{space}(\text{Deploy}(n))) \geq \text{space}(a) \)
- \( \text{remdif}\{\text{a}\}, \text{Insrt}, n \land \text{insdif}\{\text{a}\}, \text{Deploy}(n) \leftarrow \text{ADD}(a, n) \)
- ...
3 Tasks, Coalition and Reliability

3.5 Reliability

**Experiments**

**Evaluation** of three distributed algorithms in terms of:

- computation time;
- network time;
- survivability of the deployment that they find.

**Various experiment settings**

- problem’s size: number of agents + number of nodes
- number ratio: number of agents/number of nodes
- Size ratio: avg. of memory requirement of agents/avg. of memory available on nodes.

**Resulting survivability** with ASA-1 and ASA-3

**Experiment results**

- **Computation Time:** ASA-1, ASA-3 always outperform ASA-2.
- **Network Time:** ASA-1, ASA-3 much faster than ASA-2. ASA-1, ASA-3 hard to compare.
- **Survivability** of Resulting Deployments: ASA-1, ASA-3 very close. When problem size is increased, ASA-3 usually better.

See [SKZ04] for details of algorithms and experiments.

<table>
<thead>
<tr>
<th>problem size</th>
<th>n10, a8</th>
<th>n15, a12</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASA-1 CPU (microsecond)</td>
<td>113</td>
<td>236</td>
</tr>
<tr>
<td>ASA-2 CPU (microsecond)</td>
<td>218</td>
<td>420</td>
</tr>
<tr>
<td>ASA-3 CPU (microsecond)</td>
<td>164</td>
<td>310</td>
</tr>
</tbody>
</table>

| ASA-1 network time (millisecond) | 1430     |
| ASA-2 network time (millisecond) | 14291    |
| ASA-3 network time (millisecond) | 7134     |

Table 5: CPU and Network time of three algorithms with varying problem size
3.6 References


Outline

- We recapitulate very briefly sentential (also called propositional) (SL) and first-order logic (FOL).
- As an example of FOL, we consider $\text{FO}(\leq)$: monadic FOL of linear order.
- Then we present LTL, a logic to deal with linear time (no branching).
- While LTL is equivalent to $\text{FO}(\leq)$, LTL is a more compact formalism and can be easily extended.

Outline (cont.)

- $\text{CTL}^*$ is an extension of LTL to branching time.
- $\text{CTL}$ is an interesting fragment of $\text{CTL}^*$, incomparable with LTL, but with interesting computational properties.
- While LTL is defined over path formulae, $\text{CTL}$ is defined over state formulae.
- $\text{CTL}^*$ is defined over both sorts of formulae.
- We present a criterion to decide whether a $\text{CTL}^*$ formula is equivalent to a LTL formula.
4.1 Sentential Logic

Definition 4.1 (Sentential Logic $\mathcal{L}_{SL}$, Lang. $\mathcal{L} \subseteq \mathcal{L}_{SL}$)

The language $\mathcal{L}_{SL}$ of propositional (or sentential) logic consists of
- $p, q, r, x_1, x_2, \ldots, x_n, \ldots$: a countable set $\mathcal{A}$ of SL-constants,
- $\neg, \lor$: the sentential connective ($\neg$ is unary, $\lor$ is binary),
- $(, )$: the parentheses to help readability.

In most cases we consider only a finite set of SL-constants. They define a language $\mathcal{L} \subseteq \mathcal{L}_{SL}$. The set of $\mathcal{L}$-formulae $Fml_\mathcal{L}$ is defined inductively.

Macros

$\top := p \lor \neg p$
$\bot := \neg \top$

$\varphi \land \psi := \neg (\neg \varphi \lor \neg \psi)$
$\varphi \Rightarrow \psi := \neg \varphi \lor \psi$
$\varphi \leftrightarrow \psi := (\varphi \Rightarrow \psi) \land (\psi \Rightarrow \varphi)$
Definition (continued)

Thus each valuation $v$ uniquely defines a $\bar{v}$. We call $\bar{v}$ a $\mathcal{L}$-structure.

A structure determines for each formula if it is true or false. If a formula $\phi$ is true in structure $\bar{v}$ we also say $A\bar{v}$ is a model of $\phi$. From now on we will speak of models, structures and valuations synonymously.

Semantics

The process of mapping a set of $\mathcal{L}$-formulae into \{true, false\} is called semantics.

Truth Tables

Truth tables are a conceptually simple way of working with PL (invented by Wittgenstein in 1918).

<table>
<thead>
<tr>
<th>$p$</th>
<th>$\neg p$</th>
<th>$p \lor q$</th>
<th>$p \land q$</th>
<th>$p \rightarrow q$</th>
<th>$p \leftrightarrow q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>f</td>
<td>t</td>
<td>t</td>
<td>t</td>
<td>t</td>
</tr>
<tr>
<td>f</td>
<td>t</td>
<td>t</td>
<td>f</td>
<td>f</td>
<td>f</td>
</tr>
<tr>
<td>t</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>t</td>
<td>t</td>
</tr>
<tr>
<td>f</td>
<td>t</td>
<td>f</td>
<td>f</td>
<td>f</td>
<td>t</td>
</tr>
</tbody>
</table>

Definition 4.3 (Model, Theory, Tautology (Valid))

1. A formula $\varphi \in Fml_\mathcal{L}$ holds under the valuation $v$ if $\bar{v}(\varphi) = \text{true}$. We also write $\bar{v} \models \varphi$ or simply $v \models \varphi$. $\bar{v}$ is a model of $\varphi$.

2. A theory is a set of formulae: $T \subseteq Fml_\mathcal{L}$. $v$ satisfies $T$ if $\bar{v}(\varphi) = \text{true}$ for all $\varphi \in T$. We write $v \models T$.

3. A $\mathcal{L}$-formula $\varphi$ is called $\mathcal{L}$-tautology (or simply called valid) if for all possible valuations $v$ in $\mathcal{L}$ $\bar{v} \models \varphi$ holds.

From now on we suppress the language $\mathcal{L}$ when obvious from context.

Fundamental Semantical Concepts

- If it is possible to find some valuation $v$ that makes $\varphi$ true, then we say $\varphi$ is satisfiable.
- If $v \models \varphi$ for all valuations $v$ then we say that $\varphi$ is valid and write $\vdash \varphi$. $\varphi$ is also called tautology.
- A theory is a set of formulae: $\Phi \subseteq \mathcal{L}_{PL}$.
- A theory $\Phi$ is called consistent if there is a valuation $v$ with $v \models \Phi$.
- A theory $\Phi$ is called complete if for each formula $\varphi$ in the language, $\varphi \in \Phi$ or $\neg \varphi \in \Phi$.

Two simple examples

Consider the two formulae $p \land \neg b$ and $a \lor \neg a$.
- Are they satisfiable or valid?
- Are they both consistent? What if we add $b$?
Consequences

Given a theory $\Phi$ we are interested in the following question: **Which facts can be derived from $\Phi$?** We can distinguish two approaches:

1. **semantical** consequences, and
2. **syntactical** inference.

Let $\Phi$ be a theory and $\varphi$ be a formula. We say that $\varphi$ is a **semantical consequence of $\Phi$** if for all valuations $v$:

$$v \models \Phi \implies v \models \varphi.$$
Definition 4.7 (Predicate Symbols)

Let \( k \in \mathbb{N}_0 \). The set of \( k \)-ary predicate symbols (or relation symbols) is given by \( \text{Pred}^k \). Elements of \( \text{Pred}^k \) are denoted by \( P^k_1, P^k_2, \ldots \). Such a symbol takes \( k \) arguments. The set of predicate symbols is defined as

\[
\text{Pred} := \bigcup_{k} \text{Pred}^k
\]

A 0-ary predicate symbol is called (atomic) proposition.

Definition 4.8 (Term)

A term over \( \text{Func} \) and \( \text{Var} \) is inductively defined as follows:

1. Each variable from \( \text{Var} \) is a term.
2. If \( t_1, \ldots, t_k \) are terms then \( f^k(t_1, \ldots, t_k) \) is a term as well, where \( f^k \) is an \( k \)-ary function symbol from \( \text{Func}^k \).

Definition 4.9 (Language)

The first-order language with equality \( \mathcal{L}_{\text{FOL}} \) is built from terms and formulae. In the following we fix a set of variables, function-, and predicate symbols.

Definition 4.10 (Macros)

We define the following syntactic constructs as macros \( (P \in \text{Pred}^0) \):

\[
\begin{align*}
\bot & := P \land \neg P \\
\top & := \neg \bot \\
\varphi \land \psi & := \neg (\neg \varphi \lor \neg \psi) \\
\varphi \rightarrow \psi & := \neg \varphi \lor \psi \\
\varphi \leftrightarrow \psi & := (\varphi \rightarrow \psi) \land (\psi \rightarrow \varphi) \\
\forall x(\varphi) & := \neg \exists x(\neg \varphi)
\end{align*}
\]
Notation

- We will often leave out the index \( k \) in \( f^k_i \) and \( P^k_i \) indicating the arity and just write \( f_i \) and \( P_i \).
- Variables are also denoted by \( u, v, w, \ldots \)
- Function symbols are also denoted by \( f, g, h, \ldots \)
- Constants are also denoted by \( a, b, c, \ldots, c_0, c_1, \ldots \)
- Predicate symbols are also denoted by \( P, Q, R, \ldots \)

We will use our standard notation \( p \) for \( 0 \)-ary predicate symbols and also call them (atomic) propositions.

Attention

For linear temporal logic, we only need unary predicates (monadic logic) and we do not need any function symbols at all. So our terms are exactly the variables.

Semantics

Definition 4.11 (Model, Structure)

A model or structure for FOL over \( \text{Var}, \text{Func} \) and \( \text{Pred} \) is given by \( \mathcal{M} = (U, I) \) where

1. \( U \) is a non-empty set of elements, called universe or domain and
2. \( I \) is called interpretation. It assigns to each function symbol \( f^k \in \text{Func} \) a function \( I(f^k) : U^k \to U \), to each predicate symbol \( P^k \in \text{Pred} \) a relation \( I(P^k) \subseteq U^k \); and to each variable \( x \in \text{Var} \) an element \( I(x) \in U \).

We write:

1. \( \mathcal{M}(P^k) \) for \( I(P^k) \),
2. \( \mathcal{M}(f^k) \) for \( I(f^k) \), and
3. \( \mathcal{M}(x) \) for \( I(x) \).

Example 4.12

\[ \varphi := Q(x) \lor \forall z(P(x, g(z))) \lor \exists x(\forall y(P(f(x), y) \land Q(a))) \]

- \( U = \mathbb{R} \)
- \( I(\cdot) : \emptyset \to \mathbb{R}, \emptyset \to \pi \) constant functions,
- \( I(f) : I(f) = \sin : \mathbb{R} \to \mathbb{R} \) and \( I(g) = \cos : \mathbb{R} \to \mathbb{R} \),
- \( I(P) = \{(r, s) \in \mathbb{R}^2 : r \leq s \} \) and \( I(Q) = [3, \infty) \subseteq \mathbb{R} \),
- \( I(x) = \frac{\pi}{2}, I(y) = 1 \) and \( I(z) = 3 \).
Definition 4.13 (Value of a Term)

Let \( t \) be a term and \( \mathcal{M} = (U, I) \) be a model. We define inductively the value of \( t \) wrt \( \mathcal{M} \), written as \( \mathcal{M}(t) \), as follows:

- \( \mathcal{M}(x) := I(x) \) for a variable \( x = t \),
- \( \mathcal{M}(t) := I(f^k(M(t_1), \ldots, M(t_k))) \) if \( t = f^k(t_1, \ldots, t_k) \).

Example: FO(\( \leq \))

Monadic first-order logic of order, denoted by FO(\( \leq \)), is first-order logic with the only binary symbol \( \leq \) (except equality, which is also allowed) and, additionally, any number of unary predicates. The theory assumes that \( \leq \) is a linear order with least element, but nothing else.

A typical model is given by

\[
\mathcal{N} = \langle \mathbb{N}_0, \leq_{\mathbb{N}_0}, P_1^{\mathbb{N}}, P_2^{\mathbb{N}}, \ldots P_n^{\mathbb{N}} \rangle
\]

where \( \leq_{\mathbb{N}_0} \) is the usual ordering on the natural numbers and \( P_i^{\mathbb{N}} \subseteq \mathbb{N}_0 \).

The sets \( P_i^{\mathbb{N}} \) determine the timepoints where the property \( P_i \) holds.

Definition 4.14 (Semantics)

Let \( \mathcal{M} = (U, I) \) be a model and \( \varphi \in \mathcal{L}_{\text{FOL}} \). \( \varphi \) is said to be true in \( \mathcal{M} \), written as \( \mathcal{M} \models \varphi \), if the following holds:

- \( \mathcal{M} \models \varphi \) iff \( \mathcal{M}(t_1), \ldots, \mathcal{M}(t_k) \in \mathcal{M} \( P^k \) \)
- \( \mathcal{M} \models \neg \varphi \) iff not \( \mathcal{M} \models \varphi \)
- \( \mathcal{M} \models \varphi \lor \psi \) iff \( \mathcal{M} \models \varphi \) or \( \mathcal{M} \models \psi \)
- \( \mathcal{M} \models \exists x (\varphi) \) iff \( \mathcal{M}[x/a] \models \varphi \) for some \( a \in U \) where \( \mathcal{M}[x/a] \) denotes the model equal to \( \mathcal{M} \) but \( \mathcal{M}[x/a](x) = a \).
- \( \mathcal{M} \models t = r \) iff \( \mathcal{M}(t) = \mathcal{M}(r) \)

Given a set \( \Sigma \subseteq \mathcal{L}_{\text{FOL}} \) we write \( \mathcal{M} \models \Sigma \) iff \( \mathcal{M} \models \varphi \) for all \( \varphi \in \Sigma \).

What can we express in FO(\( \leq \))?

Can we find formulae that express that

- a property \( r \) is true infinitely often?
- whenever \( r \) is true, then \( s \) is true in the next timepoint?
- \( r \) is true at all even timepoints and \( \neg r \) at all odd timepoints?
Temporal logic was originally developed in order to represent tense in natural language.

Within Computer Science, it has achieved a significant role in the formal specification and verification of concurrent and distributed systems.

Much of this popularity has been achieved because a number of useful concepts can be formally, and concisely, specified using temporal logics, e.g.

- safety properties
- liveness properties
- fairness properties

Typical temporal operators:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X\varphi$</td>
<td>$\varphi$ is true in the next moment in time</td>
</tr>
<tr>
<td>$G\varphi$</td>
<td>$\varphi$ is true globally: in all future moments</td>
</tr>
<tr>
<td>$F\varphi$</td>
<td>$\varphi$ is true in finally: eventually (in the future)</td>
</tr>
<tr>
<td>$\varphi U \psi$</td>
<td>$\varphi$ becomes true until at least the moment when $\psi$ becomes true (and this eventually happens)</td>
</tr>
</tbody>
</table>

Reasoning about Time:
- The accessibility relation represents time.
- Time: linear vs. branching.
- Reasoning about a particular computation of a system.
- Models: paths (e.g. obtained from Kripke structures)
4 Linear and Branching Time
4.3 Linear Time Logic

Safety Properties

“something bad will not happen”
“something good will always hold”

Typical examples:

G¬bankrupt
GfuelOK
and so on . . .

Usually: G¬....

Liveness Properties

“something good will happen”

Typical examples:

Frich
power_on → Fonline
and so on . . .

Usually: F....

Fairness Properties

Combinations of safety and liveness possible:

FG¬dead
G(request_taxi → Farrive_taxi) → fairness

Strong fairness

“If something is requested then it will be allocated”:

G(attempt → Fsuccess),
GFattempt → GFsuccess.

Definition 4.15 (Language $L_{LTL}$ [?])

The language $L_{LTL} (\mathcal{L}_{\mathcal{P}})$ is given by all formulae generated by the following grammar, where $p \in \mathcal{P}$ is a proposition:

$$\varphi ::= p | \overline{\varphi} | \varphi \lor \varphi | \varphi U \varphi | X\varphi.$$  

The additional operators

- F (eventually in the future) and
- G (always from now on)

can be defined as macros:

$$F\varphi \equiv T U \varphi \quad \text{and} \quad G\varphi \equiv \neg F \overline{\varphi}$$

The standard Boolean connectives $\top, \bot, \land, \lor$, and $\leftrightarrow$ are defined in their usual way as macros.
Models of LTL

The semantics is given over paths, which are infinite sequences of states from $Q$, and a standard labelling function $\pi : Q \to \mathcal{P}(\text{Prop})$ that determines which propositions are true at which states.

**Definition 4.16 (Path $\lambda = q_1q_2q_3 \ldots$)**

- A path $\lambda$ over a set of states $Q$ is an infinite sequence from $Q^\omega$. We also identify it with a mapping $\mathbb{N}_0 \to Q$.
- $\lambda[i]$ denotes the $i$th position on path $\lambda$ (starting from $i = 0$) and $\lambda[i, \infty]$ denotes the subpath of $\lambda$ starting from $i$ ($\lambda[i, \infty] = \lambda[i]\lambda[i+1] \ldots$).

**Other temporal operators**

$\lambda, \pi \models F \varphi$ iff $\lambda[i, \infty], \pi \models \varphi$ for some $i \in \mathbb{N}_0$;

$\lambda, \pi \models G \varphi$ iff $\lambda[i, \infty], \pi \models \varphi$ for all $i \in \mathbb{N}_0$;

**Exercise**

Prove that the semantics does indeed match the definitions $F \varphi \equiv T \varphi$ and $G \varphi \equiv \neg F \neg \varphi$. 

**Definition 4.17 (Semantics of LTL)**

Let $\lambda$ be a path and $\pi$ be a labelling function over $Q$. The semantics of LTL, $\models_{\text{LTL}}$, is defined as follows:

- $\lambda, \pi \models_{\text{LTL}} p$ iff $p \in \pi(\lambda[0])$ and $p \in \text{Prop}$;
- $\lambda, \pi \models_{\text{LTL}} \neg \varphi$ iff not $\lambda, \pi \models_{\text{LTL}} \varphi$ (we will also write $\lambda, \pi \not\models_{\text{LTL}} \varphi$);
- $\lambda, \pi \models_{\text{LTL}} \varphi \lor \psi$ iff $\lambda, \pi \models_{\text{LTL}} \varphi$ or $\lambda, \pi \models_{\text{LTL}} \psi$;
- $\lambda, \pi \models_{\text{LTL}} X \varphi$ iff $\lambda[1, \infty], \pi \models_{\text{LTL}} \varphi$; and
- $\lambda, \pi \models_{\text{LTL}} \varphi U \psi$ iff there is an $i \in \mathbb{N}_0$ such that $\lambda[i, \infty], \pi \models \psi$ and $\lambda[j, \infty], \pi \models_{\text{LTL}} \varphi$ for all $0 \leq j < i$. 

$\lambda = q_1q_2q_3 \ldots \in Q^\omega$
4.3 Linear Time Logic

**Representation of paths**

- Paths are *infinite entities*.
- They are theoretical constructs.
- We need a *finite representation*!
- Such a finite representation is given by a transition system or a pointed Kripke structure.

$$\lambda, \pi \models GF_{pos_1} \text{ iff }$$

$$\lambda[0, \infty], \pi \models F_{pos_1} \text{ and }$$

$$\lambda[1, \infty], \pi \models F_{pos_1} \text{ and }$$

$$\lambda[2, \infty], \pi \models F_{pos_1} \text{ and }$$

...
4 Linear and Branching Time
4.3 Linear Time Logic

Some Exercises

Example 4.18

Formalise the following as LTL formulae:
1. $r$ should never occur.
2. $r$ should occur exactly once.
3. At least once $r$ should directly be followed by $s$.

Example 4.19

Formalise the following as LTL formulae:
1. $r$ is true at exactly all even states.
2. $r$ is true at each even state (the odd states do not matter). Does $r \land G(r \rightarrow XXr)$ work?

Relation to first-order logic (1)

1. The monadic first-order theory of (linear) order, $\text{FO}(\leq)$ (see Slide 425) is equivalent to LTL.
2. There is a translation from sentences of LTL to sentences of $\text{FO}(\leq)$ and vice versa, such that the LTL sentence is true in $\lambda, \pi$ iff its translation is true in the associated first-order structure.

Relation to first-order logic (2)

1. More precisely: an infinite path $\lambda$ is described as a first-order structure with domain $\mathbb{N}$ and predicates $P_p$ for $p \in \text{Prop}$. The predicates stand for the set of timepoints where $p$ is true. So each path $\lambda$ can be represented as a structure $\mathcal{N}_\lambda = \langle \mathbb{N}, \leq, P_1^{\mathbb{N}}, P_2^{\mathbb{N}}, \ldots, P_n^{\mathbb{N}} \rangle$.

   Then each LTL formula $\phi$ translates to a first-order formula $\alpha_{\phi}(x)$ with one free variable $s.t.$
   $\phi$ is true in $\lambda[n, \infty]$ iff $\alpha_{\phi}(n)$ is true in $\mathcal{N}_\lambda$.

   And conversely: for each first-order formula with a free variable there is a corresponding LTL formula s.t. the same condition holds.
The formulae GF_p, FG_p

1. What are their counterparts in FO(≤)?
2. We will see later that FG_p does not belong to CTL, but to CTL*. It is not even equivalent to a CTL formula.
3. However, GF_p is equivalent to a CTL formula: AGAF_p

Some Remarks

1. A particular logic LTL is determined by the number n of propositional variables. Strictly speaking, this number should be a parameter of the logic. This also applies to the logics CTL and ATL.
2. While both F and G can be expressed using U, the converse is not true: U can not be expressed by F and G.

Satisfiability of LTL formulae

A formula is satisfiable, if there is a path where it is true. Can we restrict the structure of such paths? I.e. can we restrict to simple paths, for example paths that are periodic?

- If this is the case, then we might be able to construct counterexamples more easily, as we need only check very specific paths.
- It would be also useful to know how large the period is and within which initial segment of the path it starts, depending on the length of the formula ϕ.

Satisfiability of LTL formulae (cont.)

Theorem 4.20 (Periodic model theorem [?])

A formula ϕ ∈ L_{LTL} is satisfiable iff there is a path λ which is ultimately periodic, and the period starts within 2^{1+|ϕ|} steps and has a length which is ≤ 4^{1+|ϕ|}.
4.4 Branching Time Logic

- **Path quantifiers**: \( A \) (for all paths), \( E \) (there is a path);
- **Temporal operators**: \( X \) (nexttime), \( F \) (finally), \( G \) (globally) and \( U \) (until);
- **CTL**: each temporal operator must be immediately preceded by exactly one path quantifier;
- **CTL\(^*\)**: no syntactic restrictions.

**Example 4.21 (Branching Time)**

In this structure, whenever \( p \) holds at some timepoint, then there is a path where \( q \) holds in the next step and there is (another) path where \( \neg q \) holds in the next step. And this holds along all paths (there are three infinite paths).
Definition 4.22 ($\mathcal{L}_{\text{CTL}^*}$)

The language $\mathcal{L}_{\text{CTL}^*}(\text{Prop})$ is given by all formulae generated by the following grammar:

$$\varphi ::= p \mid \neg \varphi \mid \varphi \lor \varphi \mid E \gamma$$

where

$$\gamma ::= \varphi \mid \neg \gamma \mid \gamma \lor \gamma \mid \gamma U \gamma \mid X \gamma$$

and $p \in \text{Prop}$. Formulae $\varphi$ (resp. $\gamma$) are called state (resp. path) formulae.

We use the same abbreviations as for $\mathcal{L}_{\text{LTL}}$

- $\lambda, \pi \models F \varphi$ iff $\lambda[i, \infty], \pi \models \varphi$ for some $i \in \mathbb{N}_0$;
- $\lambda, \pi \models G \varphi$ iff $\lambda[i, \infty], \pi \models \varphi$ for all $i \in \mathbb{N}_0$;

For example, $A G E X p$ is a $\mathcal{L}_{\text{CTL}^*}$-formula whereas $A G F p$ is not.

Example 4.24 ($\text{CTL}^*$ or $\text{CTL}$?)

Are the following $\text{CTL}^*$ or $\text{CTL}$ formulae? What do they express?

1. $EF \neg \text{shutdown}$
2. $EF \neg \text{shutdown}$
3. $AG \neg \text{rain}$
4. $AG \neg \text{rain}$ (Is it different from (3)?)
5. $EFG \neg \text{broken}$
6. $AG(p \rightarrow (EX q \land EX \neg q))$
The precise definition of Kripke structures is given in the next section. To understand the following definitions it suffices to note that:

- Given a set of states $Q$ (each is a propositional model), a Kripke model $M$ is simply a tuple $(Q, R)$ where $R \subseteq Q \times Q$ is a binary relation.
- $q_1 R q_2$ (also written $(q_1, q_2) \in R$) means that state $q_2$ is reachable from state $q_1$ (by executing certain actions).
- The relation $R$ is serial: for all $q$ there is a $q'$ such that $q R q'$. This ensures that our paths are infinite.
- Given a state $q$ in a Kripke model, by $\Lambda(q)$ we mean the set of all paths determined by the relation $R$ starting in $q$: $q, q_1, q_2, \ldots, q_i, \ldots$ where $q R q_1, \ldots q_i R q_{i+1}, \ldots$

Definition 4.25 (Semantics $\models_{\text{CTL}^\ast}$)

Let $M$ be a Kripke model, $q \in Q$ and $\lambda \in \Lambda$. The semantics of $L_{\text{CTL}^\ast}$- and $L_{\text{CTL}}$-formulae is given by the satisfaction relation $\models_{\text{CTL}^\ast}$ for state formulae by:

- $M, q \models_{\text{CTL}^\ast} \phi$ iff $\lambda[0] \in \pi(p)$ and $p \in \text{Prop}$;
- $M, q \models_{\text{CTL}^\ast} \neg \phi$ iff $M, q \not\models_{\text{CTL}^\ast} \phi$;
- $M, q \models_{\text{CTL}^\ast} \phi \lor \psi$ iff $M, q \models_{\text{CTL}^\ast} \phi$ or $M, q \models_{\text{CTL}^\ast} \psi$;
- $M, q \models_{\text{CTL}^\ast} \text{Ex} \phi$ iff there is a path $\lambda \in \Lambda(q)$ such that $M, \lambda \models_{\text{CTL}^\ast} \phi$;
- $M, q \models_{\text{CTL}^\ast} \text{EG} \phi$ iff there is a path $\lambda \in \Lambda(q)$ such that $M, \lambda[i] \models_{\text{CTL}^\ast} \phi$ for every $i \geq 0$;
- $M, q \models_{\text{CTL}^\ast} \phi \Upsilon $ iff there is a path $\lambda \in \Lambda(q)$ such that $M, \lambda[j] \models_{\text{CTL}^\ast} \phi$ for some $i \geq 0$, and $M, \lambda[j] \models_{\text{CTL}^\ast} \phi$ for all $0 \leq j < i$.

State-based semantics for CTL

- $M, q \models_{\text{CTL}} p$ iff $q \in \pi(p)$;
- $M, q \models_{\text{CTL}} \neg \phi$ iff $M, q \not\models_{\text{CTL}^\ast} \phi$;
- $M, q \models_{\text{CTL}} \phi \lor \psi$ iff $M, q \models_{\text{CTL}^\ast} \phi$ or $M, q \models_{\text{CTL}^\ast} \psi$;
- $M, q \models_{\text{CTL}} \text{Ex} \phi$ iff there is a path $\lambda \in \Lambda(q)$ such that $M, \lambda[i] \models_{\text{CTL}^\ast} \phi$;
- $M, q \models_{\text{CTL}} \text{EG} \phi$ iff there is a path $\lambda \in \Lambda(q)$ such that $M, \lambda[i] \models_{\text{CTL}^\ast} \phi$ for every $i \geq 0$;
- $M, q \models_{\text{CTL}} \phi \Upsilon $ iff there is a path $\lambda \in \Lambda(q)$ such that $M, \lambda[j] \models_{\text{CTL}^\ast} \phi$ for some $i \geq 0$, and $M, \lambda[j] \models_{\text{CTL}^\ast} \phi$ for all $0 \leq j < i$.

Is this complicated semantics over paths necessary for CTL?
LTL as subset of CTL\(^\ast\)

LTL is interpreted over infinite chains (infinite words), but not over (serial) Kripke structures (which are branching).

- To consider LTL as a subset of CTL\(^\ast\), one can just add the quantifier A in front of a LTL formula and use the semantics of CTL\(^\ast\). For infinite chains, this semantics coincides with the LTL semantics.

- The theorem of Clarke und Draghiescu gives a nice characterization of those CTL\(^\ast\) formulae that are equivalent to LTL formulae. Given a CTL\(^\ast\) formula \(\varphi\), we construct \(\varphi'\) by just forgetting all path operators. Then

  \(\varphi\) is equivalent to a LTL formula
  iff
  \(\varphi\) and \(A\varphi'\) are equivalent under the semantics of CTL\(^\ast\).

Application of Clarke and Draghiescu

We consider the LTL formula \(\text{GF} p\). Viewed as a CTL\(^\ast\) formula it becomes \(A \text{GF} p\). But this is equivalent (in CTL\(^\ast\)) to \(A \text{AF} p\), a CTL formula.

Now we consider the CTL formula \(E \text{GE} p\). It is not equivalent to any LTL formula. This is because \(E \text{GE} p\) and \(A \text{GF} p\) are not equivalent in CTL\(^\ast\):

\[
\begin{array}{c}
q_0 \\
q_1 \\
q_2
\end{array}
\]

The first formula holds, the second does not.

Example 4.26 (Robots and Carriage)

- Two robots push a carriage from opposite sides.
- Carriage can move clockwise or anticlockwise, or it can remain in the same place.
- 3 positions of the carriage.
- We label the states with propositions \(\text{pos}_0\), \(\text{pos}_1\), \(\text{pos}_2\), respectively, to allow for referring to the current position of the carriage in the object language.
4 Linear and Branching Time
4.4 Branching Time Logic

Figure 30: Two robots and a carriage: A schematic view (left) and a transition system $M_0$ that models the scenario (right).

Example: Rocket and Cargo

- A rocket and a cargo.
- The rocket can be moved between London (proposition $roL$) and Paris (proposition $roP$).
- The cargo can be in London ($caL$), Paris ($caP$), or inside the rocket ($caR$).
- The rocket can be moved only if it has its fuel tank full ($fuelOK$).
- When it moves, it consumes fuel, and $nofuel$ holds after each flight.

It becomes more interesting if abilities of agents are considered $\rightsquigarrow$ ATL.
4 Linear and Branching Time
4.4 Branching Time Logic

Example: Rocket and Cargo

In our logics, we assumed a serial accessibility relation: no deadlocks are possible.
One can also allow states with no outgoing transitions. In that case, in the semantical definition of $E$ on Slide 461 one has to replace “there is a path” by “there is an infinite path or one which can not be extended”.
Similar modifications are needed in the definition of CTL.
One can also add to each state with no outgoing transitions a special transition leading to a new state that loops into itself.

How to express that there is no possibility of a deadlock?

$$AGX\top$$ ($\dashv\vdash$ CTL$^*$)    $$AG\neg X\top$$ ($\dashv\vdash$ CTL)
We first introduce a calculus for propositional logic that is sound and complete: this is a result that we need later in order to develop sound and complete axiomatic systems for modal and temporal logics (Section 5.1).

We then extend propositional logic by modality operators. This allows to express notions like beliefs, knowledge, and time (Section 5.2).

We introduce the basic semantics of modal logics (ML), based on Kripke-structures and state several interesting axioms (Section 5.3).

We then consider epistemic logics in (Section 5.4).

One extension of ML is dynamic logic: we add program/action modalities \([\alpha], \langle \alpha \rangle\), to talk about actions and their outcomes. This is the basis for developing programming languages for MAS (Section 5.5).

In Chapter 4 we have introduced the temporal logics LTL and CTL. They are instances of modal logic: for each logic, we are interested in the formulae true in a particular class of Kripke models.

A natural question is can we axiomatize these models by suitable formulae?

We give axiomatizations for LTL, CTL, and ATL (Section 5.6).

5.1 A calculus for propositional logic

**Definition 5.1 (Hilbert-Type Calculi)**

A Hilbert-Type calculus over a language \( \mathcal{L} \) is a pair \((\text{Ax}, \text{Inf})\) where

- \( \text{Ax} \): is a subset of \( Fml_{\mathcal{L}} \), the set of well-formed formulae in \( \mathcal{L} \): they are called axioms,

- \( \text{Inf} \): is a set of pairs written in the form

\[
\frac{\phi_1, \phi_2, \ldots, \phi_n}{\psi}
\]

where \( \phi_1, \phi_2, \ldots, \phi_n, \psi \) are \( \mathcal{L} \)-formulae: they are called inference rules.

Intuitively, one can assume all axioms as “true formulae” (tautologies) and then use the inference rules to derive even more new formulae.
**Definition 5.2 (Calculus for Sentential Logic SL)**

We define \( \text{Hilbert}_{SL} = \langle \text{Ax}_{SL}^L, \{\text{MP}\} \rangle \), the Hilbert-Type calculus: \( L \subseteq L_{SL} \) with the wellformed formulae \( Fml_{SL} \) as defined in Definition 4.1.

Axioms in SL (\( \text{Ax}_{SL}^L \)) are the following formulae:

1. \( \phi \rightarrow \top \), \( \bot \rightarrow \phi \), \( \neg \top \rightarrow \bot \), \( \bot \rightarrow \neg \top \),
2. \( (\phi \rightarrow \psi) \rightarrow ((\phi \rightarrow (\psi \rightarrow \chi)) \rightarrow (\phi \rightarrow \chi)) \),
3. \( (\phi \land \psi) \rightarrow \phi, (\phi \land \psi) \rightarrow \psi \),
4. \( \phi \rightarrow (\phi \lor \psi), \psi \rightarrow (\phi \lor \psi) \),
5. \( \neg \neg \phi \rightarrow \phi, (\phi \rightarrow \psi) \rightarrow ((\phi \rightarrow \neg \psi) \rightarrow \neg \phi) \),
6. \( \phi \rightarrow (\psi \rightarrow \phi), \phi \rightarrow (\psi \rightarrow (\phi \land \psi)) \),
7. \( (\phi \rightarrow \chi) \rightarrow ((\psi \rightarrow \chi) \rightarrow (\phi \lor \psi \rightarrow \chi)) \).

\( \phi, \psi, \chi \) stand for arbitrarily complex formulae (not just constants). They represent schemata, rather than formulae in the language.

---

**Definition (continued)**

The only inference rule in SL is modus ponens:

\[
\text{MP} : Fml \times Fml \rightarrow Fml : (\varphi, \varphi \rightarrow \psi) \rightarrow \psi.
\]

or short

\[
(\text{MP}) \quad \varphi, \varphi \rightarrow \psi.
\]

(\( \varphi, \psi \) are arbitrarily complex formulae).

---

**Definition 5.3 (Proof)**

A proof of a formula \( \varphi \) from a theory \( T \subseteq Fml_{L} \) is a sequence \( \varphi_1, \ldots, \varphi_n \) of formulae such that \( \varphi_n = \varphi \) and for all \( i \) with \( 1 \leq i \leq n \) one of the following conditions holds:

- \( \varphi_i \) is substitution instance of an axiom,
- \( \varphi_i \in T \),
- there is \( \varphi_l, \varphi_k = (\varphi_l \rightarrow \varphi_i) \) with \( l, k < i \). Then \( \varphi_i \) is the result of the application of modus ponens on the predecessor-formulae of \( \varphi_i \).

We write: \( T \vdash \varphi \) (\( \varphi \) can be derived from \( T \)).

---

**We show that:**

1. \( \vdash \top \) and \( \vdash \bot \).
2. \( A \vdash A \lor B \) and \( \vdash A \lor \neg A \).
3. The rule

\[
(\text{R}) \quad A \rightarrow \varphi, \neg A \rightarrow \psi \quad \varphi \lor \psi
\]

can be derived.

- Our version of sentential logic does not contain a connective “\( \leftrightarrow \)”. We define “\( \varphi \leftrightarrow \psi \)” as a macro for “\( \varphi \rightarrow \psi \land \psi \rightarrow \phi \)”. Show the following:

  If \( \vdash \varphi \leftrightarrow \psi \), then \( \vdash \varphi \) if and only if \( \vdash \psi \).
We have now introduced two important notions:

**Syntactic derivability** \( \vdash \): the notion that certain formulae can be derived from other formulae using a certain calculus,

**Semantic validity** \( \models \): the notion that certain formulae follow from other formulae based on the semantic notion of a model.

**Definition 5.4 (Correct-, Completeness for a calculus)**

Given an arbitrary calculus (which defines a notion \( \vdash \)) and a semantics based on certain models (which defines a relation \( \models \)), we say that

**Correctness**: The calculus is *correct* with respect to the semantics, if the following holds:

\[ \Phi \vdash \phi \text{ implies } \Phi \models \phi. \]

**Completeness**: The calculus is *complete* with respect to the semantics, if the following holds:

\[ \Phi \models \phi \text{ implies } \Phi \vdash \phi. \]

**Theorem 5.5 (Correct-, Completeness for Hilbert \( \text{SL}_L \))**

A formula follows semantically from a theory \( T \) if and only if it can be derived:

\[ T \models \varphi \text{ if and only if } T \vdash \varphi. \]

**Theorem 5.6 (Compactness for Hilbert \( \text{SL}_L \))**

A formula follows from a theory \( T \) if and only if it follows from a finite subset of \( T \):

\[ Cn(T) = \bigcup \{ Cn(T') : T' \subseteq T, T' \text{finite} \}. \]
5.2 Basic Modal Logic

What is a Logic?

We present a framework for thinking about logics as:
- languages for describing a problem,
- ways of talking about relational structures and models.

These are the two key components in the way we will approach logic:

1. **Language:** fairly simple, precisely defined, formal languages.
2. **Model** (or relational structure):
   simple “world” that the logic talks about.

Various modal logics

- knowledge $\rightarrow$ epistemic logic (\(\hookrightarrow\) Section 4.4),
- beliefs $\rightarrow$ doxastic logic,
- obligations $\rightarrow$ deontic logic,
- actions $\rightarrow$ dynamic logic (\(\hookrightarrow\) Section 4.5),
- time $\rightarrow$ temporal logic (\(\hookrightarrow\) Section 4),
- and combinations of the above.

Most famous multimodal logics: BDI logics of beliefs, desires, intentions (and time).

Relational Structures

**Definition 5.7 (Relational Structure)**

A relational structure is given by \((W, \{R_1, \ldots, R_n\})\) and consists of:
- A non-empty set \(W\), the elements of which are our objects of interest. They are called points, states, nodes, worlds, times, instants or situations.
- A non-empty set \(\{R_1, \ldots, R_n\}\) of relations, \(R_i \subseteq W \times W\).
The Basic Modal Language

- Standard propositional logic can be seen as a one-point relational structure.
- But relational structures can describe much more. We can talk about points, lines etc.
- Therefore, we introduce the basic modal language.

We build the basic modal language on top of the propositional language by extending $\mathcal{L}_{PL}(\text{Prop})$ with two new operators:

### Possibility and necessity

- $\Diamond \varphi$: $\varphi$ is possible
  (We see one or more states where $\varphi$ holds.)
- $\Box \varphi$: $\varphi$ is necessary
  (In all reachable states $\varphi$ holds.)

Boolean macros are defined in the standard way. Additionally, we have the dual $\Box$ (called “box”) of $\Diamond$:

$$\Box \varphi := \neg \Diamond \neg \varphi$$

Example formulae: $\Box \neg p$, $\text{FF} \varphi \land \Box \text{F} \psi$

We can talk about attributes by adding labels to nodes.

**Example 5.9 (Colored graph I)**

Imagine standing in a node of a colored graph. What can we see?

- $\Diamond \text{blue}$
- $(\text{black} \land \text{red}) \land \Diamond \Diamond \text{green}$
5 From Modal to Dynamic Logic

5.2 Basic Modal Logic

Example 5.11

Colored graph II

blue \rightarrow \square \text{black}
green \rightarrow \square \text{black}
yellow \rightarrow \diamond \text{yellow}

Definition 5.12 (Kripke frame)
A Kripke frame is given by $\mathcal{F} = (W; \mathcal{R})$ where
- $W$ is a non-empty set, called set of domains or worlds,
- $\mathcal{R} \subseteq W \times W$ is a binary relation.

Frames are mainly used to talk about validities: They stand for a whole set of models.

Definition 5.13 (Kripke model)
A Kripke model is given by $\mathcal{M} = (W, \mathcal{R}, \mathcal{V})$ where
- $(W, \mathcal{R})$ is a Kripke frame,
- $\mathcal{V}: \text{Prop} \rightarrow \mathcal{P}(W)$ is called labelling function or valuation.
We also use $\mathcal{V}: W \rightarrow \mathcal{P}(\text{Prop})$.

Kripke frames (resp. models) are simply relational structures (resp. with labels!)

Example 5.14

Consider the frame $\mathcal{F} = (\{w_1, w_2, w_3, w_4, w_5\}; \mathcal{R})$ where
$\mathcal{R} w_i w_j \text{ iff } j = i + 1$ and $\mathcal{V}(p) = \{w_2, w_3\}$,
$\mathcal{V}(q) = \{w_1, w_2, w_3, w_4, w_5\}$, $\mathcal{V}(r) = \emptyset$.

Frames vs. Models?

Frames

Mathematical pictures of ontologies that we find interesting. That is, frames define the fundamental structure of the domain of interest.

For example, we model time as a collection of points ordered by a strict partial order.

Models

Frames are extended by contingent information. That is, models extend the mathematical structure provided by frames by additional information.
5 From Modal to Dynamic Logic
5.2 Basic Modal Logic

Formal semantics of $\mathcal{L}_{ML}$.

**Definition 5.15 (Semantics $M, w \models \varphi$)**

Let $M$ be a Kripke model, $w \in W_M$, and $\varphi \in \mathcal{L}_{ML}$. $\varphi$ is said to be locally true or satisfied in $M$ and world $w$, written as $M, w \models \varphi$, if the following holds:

- $M, w \models p$ iff $w \in V_M(p)$ and $p \in \text{Prop}$,
- $M, w \models \neg \varphi$ iff not $M, w \models \varphi$,
- $M, w \models \varphi \lor \psi$ iff $M, w \models \varphi$ or $M, w \models \psi$,
- $M, w \models \varphi$ iff there is a world $w' \in W$ such that $Rww'$ and $M, w' \models \varphi$.

Given a set $\Sigma \subseteq \mathcal{L}_{ML}$ we write $M, w \models \Sigma$ iff $M, w \models \varphi$ for all $\varphi \in \Sigma$.

What about $\Box \varphi$? $\sim$ blackboard

---

Some Examples

**Example 5.16**

$\mathcal{F} = (\{w_1, w_2, w_3, w_4, w_5\}, R)$ where $Rw_jw_i$ iff $j = i + 1$ and $V(p) = \{w_2, w_3\}$, $V(q) = \{w_1, w_2, w_3, w_4, w_5\}$, $V(r) = \emptyset$.

\[
\begin{array}{cccccc}
& & & & & \\
& & q & p & q & q & q \\
& & w_1 & w_2 & w_3 & w_4 & w_5
\end{array}
\]

1. $M, w_1 \models F\Box p$
2. $M, w_1 \not\models F\Box p \rightarrow p$
3. $M, w_2 \models F(p \land \neg r)$
4. $M, w_1 \models q \land F(q \land F(q \land Fq)))$
5. $M \models \Box q$

---

Validity and (Global) Satisfaction

We take on a global point of view.

Given a specification like $\varphi := \Box \neg \text{crash}$. In which states should it be true?

**Definition 5.17 (Validity)**

A formula $\varphi$ is called valid or globally true in a model $M$ iff $M, w \models \varphi$ for all $w \in W_M$. We write $M \models \varphi$.

$\varphi$ is satisfiable in $M$ if $M, w \models \varphi$ for some $w \in W_M$.

Analogously, we say that a set $\Sigma$ of formulae is valid (resp. satisfiable) in $M$ iff all formulae in $\Sigma$ are valid (resp. satisfiable) in $M$.

Validity and satisfiability are dual concepts!
**Example 5.18**

In which models is the following formula true?

\[ \Box(p \rightarrow q) \rightarrow (\Box p \rightarrow \Box q) \]

\[ M, w = \Box(p \rightarrow q) \]

iff \( \forall w' (wRw' \Rightarrow M, w' = p \rightarrow q) \)

implies \( \forall w' (wRw' \Rightarrow M, w' = p) \Rightarrow \)

iff \( M, w = \Box p \Rightarrow M, w = \Box q \)

iff \( M, w = \Box p \rightarrow \Box q \)

The formula is true in any frame and hence in any model.

---

**Frames and Validity**

In Example 5.18 we have seen that a formula can be true/false for all valuations. We can also ignore contingent information.

**Definition 5.20 (Frame Validity: \( \mathfrak{F} \models \varphi \))**

Let \( \mathfrak{F} \) be a frame and \( \varphi \in \mathcal{L}_{\text{BML}} \).

1. \( \varphi \) is valid in \( \mathfrak{F} \) and \( w \in W_{\mathfrak{F}} \), written \( \mathfrak{F}, w \models \varphi \) , if

\[ M, w = \varphi \text{ for all models } M = (\mathfrak{F}, \pi) \text{ based on } \mathfrak{F}. \]

2. \( \varphi \) is valid in \( \mathfrak{F} \), written \( \mathfrak{F} \models \varphi \), if \( \mathfrak{F}, w \models \varphi \) for all \( w \in W_{\mathfrak{F}} \).

3. Let \( \mathcal{F} \) be class of frames. \( \varphi \) is said to be valid in \( \mathcal{F} \), if \( \varphi \) is valid in each frame \( \mathfrak{F} \in \mathcal{F} \).

---

**Modal Consequence Relation**

Up to now we verified formulae in a given model and state. Often, it is interesting to know whether a property follows from a given set of formulae.

**Definition 5.19 (Local Consequence Relation)**

Let \( M \) be a class of models, \( \Sigma \) be a set of formulae and \( \varphi \) be a formula.

- \( \varphi \) is a (local) semantic consequence of \( \Sigma \) over \( M \), written \( \Sigma \models_M \varphi \), if for all \( M, w \in W_M \) it holds that

\[ M, w = \varphi. \]

- If \( M \) is the class of all models we just say that \( \varphi \) is a (local) consequence of \( \Sigma \) and write \( \Sigma \models \varphi \).

---

**Lemma 5.21 (Distribution Axioms)**

The two formulae

\[ F(p \lor q) \rightarrow (Fp \lor Fq) \]

\[ (p \rightarrow q) \rightarrow (\Box p \rightarrow \Box q) \]

are both valid in all Kripke frames \( \mathfrak{F} \). The last formula is also called axiome K.

**Proof.**

\( \sim \) Exercise and Example 5.18.
5 From Modal to Dynamic Logic

5.2 Basic Modal Logic

Example 5.22
Is $\mathcal{F} \top$ valid in all frames? In which class is the formula valid?

What about $\Box \top$? ~ blackboard

Example 5.23
Is $\mathcal{F} \mathcal{F} p \rightarrow \mathcal{F} p$ true in $w_1$?

Is there a class of frames in which formula is valid? ~ blackboard

Example 5.24
Let $\mathcal{M}$ be the class of transitive models. Then:

1. $\lozenge \lozenge p \models \mathcal{M} \lozenge p$.
2. $\Box p \models \mathcal{M} \Box \Box p$, but
3. $\Box \Box p \models \mathcal{M} \Box p$ does not hold.

Is there a class of models $\mathcal{M}$ for which $\lozenge \lozenge p \models \mathcal{M} \lozenge p$ holds, but no model in $\mathcal{M}$ is transitive?

Yes!

5.3 Correspondence Theory
Correspondence Theory

We consider more axioms systematically.

Definition 5.25 (KDT45)

We assume that we have one modal operator $\Box$.

$\mathbf{K}$ $\Box(p \rightarrow q) \rightarrow (\Box p \rightarrow \Box q)$

$\mathbf{D}$ $\neg \Box(p \land \neg p)$

$\mathbf{T}$ $\Box p \rightarrow p$

$\mathbf{4}$ $\Box p \rightarrow \Box \Box p$

$\mathbf{S}$ $\neg \Box p \rightarrow \Box \neg \Box p$

Properties of Frame (1)

Properties of the accessibility relations $\mathcal{R}$ of frames:

- **Serial**: For all $w$ there is a $w'$ with $w \mathcal{R} w'$.
- **Reflexive**: For all $w$: $w \mathcal{R} w$.
- **Transitive**: For all $w, w', w'': w \mathcal{R} w'$ and $w' \mathcal{R} w''$ implies $w \mathcal{R} w''$.
- **Euclidean**: For all $w, w', w'': w \mathcal{R} w'$ and $w \mathcal{R} w''$ implies $w' \mathcal{R} w''$.
- **Symmetric**: For all $w, w'$: $w \mathcal{R} w'$ implies $w' \mathcal{R} w$.

Example 5.27

We have

$\mathcal{G} \models \Box p \rightarrow p$ iff $\mathcal{G}$ is reflexive.

Let $\mathcal{G}$ be a frame satisfying $\Box p \rightarrow p$. That is,

for all $w \in W$, $\mathcal{G}, w \models \Box p \rightarrow p$.

This is the case, if for all models $\mathcal{M}$ over $\mathcal{G}$ and

for all $w \in W$, $\mathcal{M}, w \models \Box p \rightarrow p$.

Which properties must $\mathcal{R}$ satisfy? Suppose $\mathcal{R}$ is not reflexive. Then, there is a state $w'$ with not $w' \mathcal{R} w'$. Make $p$ true at all states of $W \setminus \{w'\}$. Then, $\mathcal{M}, w' \not\models \Box p \rightarrow p$ and hence $\mathcal{G} \not\models \Box p \rightarrow p$. **Contradiction!**
Now suppose we are given a reflexive frame $\mathfrak{F}$ and suppose $\mathfrak{F} \not\models \Box p \rightarrow p$.

- Then, there is a model $\mathcal{M} = (\mathfrak{F}, \pi)$ and a state $w$, $\mathcal{M}, w \models \Box p$ and $\mathcal{M}, w \not\models p$.
- By reflexivity we have $wRw$.
- But then, from $\mathcal{M}, w \models \Box p$ it follows that $\mathcal{M}, w \models p$.
- Contradiction!

We must have $\mathfrak{F} \models \Box p \rightarrow p$.

In other words, axiom $T$ characterises reflexive frames.

Axiomatic Systems

As in classical logic, one can ask about a complete axiom system. Is there a calculus that allows to derive all sentences true in all Kripke models?

Definition 5.29 (System K)

System $K$ is an extension of propositional logic by the axiom

$$K \ (\Box \varphi \land \Box (\varphi \rightarrow \psi)) \rightarrow \Box \psi$$

and the inference rule $\frac{\varphi}{\Box \varphi}$ (Necessitation).

We also need the duality axiom $\Box \varphi \leftrightarrow \neg \Diamond \neg \varphi$ (as we introduced $\Box$ as macro).

Note, $\varphi$ and $\psi$ can be substituted by any formula.
Example ctd.

\[ \vdash K (\Box p \land \Box q) \rightarrow \Box (p \land q) \]

1 \[ \vdash p \rightarrow (q \rightarrow (p \land q)) \quad \text{(prop. tautology)} \]
2 \[ \vdash \Box (p \rightarrow (q \rightarrow (p \land q))) \quad \text{(necessitation)} \]
3 \[ \vdash \Box (\varphi \rightarrow \psi) \rightarrow (\Box \varphi \rightarrow \Box \psi) \quad \text{(axiom K)} \]
4 \[ \vdash \Box (p \rightarrow (q \rightarrow (p \land q))) \rightarrow (\Box p \rightarrow \Box (p \rightarrow (p \land q))) \quad \text{(subst. 3, } \varphi = p, \psi = p \rightarrow (p \land q)) \]
5 \[ \vdash \Box p \rightarrow \Box (q \rightarrow (p \land q)) \quad \text{(MP 2, 3)} \]
6 \[ \vdash \Box (q \rightarrow (p \land q)) \rightarrow (\Box q \rightarrow \Box (p \land q)) \quad \text{(subst. 3, } p = q, q = p \land q) \]
7 \[ \vdash \Box p \rightarrow (\Box q \rightarrow \Box (p \land q)) \quad \text{(prop. logic 5, 6)} \]
8 \[ \vdash (\Box p \land \Box q) \rightarrow \Box (p \land q) \quad \text{(prop. logic 7)} \]

Theorem 5.32 (Sound-/completeness of K)

System K is sound and complete with respect to arbitrary Kripke models.

- Note that we have not assumed any properties of the accessibility relation \( R \): It is just any binary relation.
- Assuming that \( R \) is an equivalence relation, what additional statements (axioms) are true in all such Kripke models?

Theorem 5.33 (Sound/complete subsystems)

Let \( X \) be any subset of \{D, T, 4, 5, B\} and let \( X' \) be the subset of \{serial, reflexive, transitive, euclidean, symmetric\} corresponding to \( X \).

Then system K extended with axioms \( X \) is sound and complete with respect to Kripke frames which satisfy properties \( X' \).

Corollary 5.34 (KT45)

System KT45 is sound and complete with respect to Kripke frames with an accessibility relation which is an equivalence relation.

Example 5.35

KT45 and KTD45 are both sound and complete with respect to Kripke frames in which the accessibility relation is an equivalence relation.

What does that mean for the properties corresponding to these axioms?

Any reflexive, transitive and euclidean relation is also serial!
5.4 Epistemic Logic

Interpreting □, as knowledge

Let us now assume we have several agents $i$ and we interpret $\Box_i \varphi$ as agent $i$ knows that $\varphi$. In that case one often writes $K_i \varphi$ instead of $\Box_i \varphi$.

Accessibility relation

What does the equivalence relation encode? Incomplete information:

$w \sim w'$ $\Rightarrow$ The agent cannot distinguish $w$ and $w'$. Both states provide the same information.

Knowledge = Truth in all indistinguishable states

What other properties should hold when interpreting $\Box$ as knowledge?

$K \quad K(p \rightarrow q) \rightarrow (Kp \rightarrow Kq)$

$D \quad \neg K \bot \quad \sim$ consistency

$T \quad Kp \rightarrow p \quad \sim$ truth

$4 \quad Kp \rightarrow KKp \quad \sim$ positive introspection

$S \quad \neg Kp \rightarrow K\neg Kp \quad \sim$ negative introspection

Muddy children

Example 5.36 (Muddy children)

- A group of playing children is called back by their father. They gather around him.
- Some of them have become dirty:
  1. they may have mud on their forehead,
  2. children can only see whether others are muddy,
  3. and not if there is any mud on their own forehead.
- All this is commonly known, and the children are perfect logicians.
Now the father announces the following:
- Father: “At least one of you has mud on his or her forehead.”
- Father: “Will those who know whether they are muddy please step forward.”
- If nobody steps forward, father keeps repeating the request.

**Question**

What happens?
5 From Modal to Dynamic Logic

5.4 Epistemic Logic

\[ \mathcal{M}, q_{\text{mmcm}} \models m_1 \land m_2 \land \neg m_3 \]
\[ \mathcal{M}, q_{\text{mmcm}} \models \neg K_1 m_1 \land K_1 m_2 \]
\[ \mathcal{M}, q_{\text{mmcm}} \models K_1 K_2 m_2 \land K_1 \neg K_2 m_2 \]

Father: “At least one is muddy.”

\[ \neg K_1 m_1 \land \neg K_2 m_2 \land \neg K_3 m_3 \]
Father (1): “If you know that you’re muddy, raise your hand.”
Nothing happens.

Father (2): “If you know that you’re muddy, raise your hand.”
The kids see that nobody has raised their hands after (1)!
Children with mud can eliminate worlds...

Interpreting □ as belief

Up to now we were thinking of □ᵢ as agent i knows that ϕ.
What if we interpret the operator as belief?

Under such an interpretation axiom T is usually not assumed to hold. But all other axioms make sense.

Definition 5.37 (System KD45)

Axiom system KD45 is called the standard logic of beliefs.
Axiom K is called logical omniscience, axiom D is called consistency, axiom 4 (resp. axiom 5) is called positive (resp. negative) introspection.
5.5 Dynamic Logic

1st idea: Consider actions or atomic programs $\alpha$. Each such $\alpha$ defines a transition (accessibility relation) from worlds into worlds.

2nd idea: We need statements about the outcome of actions:
- $[\alpha]\varphi$: “after each execution of $\alpha$, $\varphi$ holds,
- $\langle\alpha\rangle\varphi$: “after some executions of $\alpha$, $\varphi$ holds.

As usual, $\langle\alpha\rangle\varphi \equiv \neg[\alpha]\neg\varphi$.

3rd idea: Programs/actions can be combined (sequentially, nondeterministically, iteratively), e.g.:

$[\alpha; \beta]\varphi$

would mean “after each execution of $\alpha$ and then $\beta$, formula $\varphi$ holds”.

Can we combine these three ideas and come up with a language and logic where we can express all these features?

Dynamic Logic over arbitrary programs

**Example 5.38 (Propositional Dynamic Logic)**

Infinite collection of diamonds: $D_p = \{\pi \mid \pi \text{ is a program}\}$

What do the following operators express?

- $\langle\pi\rangle\varphi$: Some terminating execution of $\pi$ leads to a state with information $\varphi$
- $[\pi]\varphi$: Each terminating execution of $\pi$ leads to a state with information $\varphi$

It would be nice if we could combine simple programs:

- $\pi \cup \pi'$: Nondeterministic choice
- $\pi; \pi'$: Sequential composition
- $\pi^*$: Iterative execution
What do the following statements express?

\[ (\pi^*)\varphi \leftrightarrow \varphi \lor (\pi; \pi^*)\varphi : \text{A state with information } \varphi \text{ is reached by executing } \pi \text{ a finite number of times iff the current state satisfies } \varphi \text{ or we can execute } \pi \text{ once and reach a state in which } \varphi \text{ holds by executing } \pi \text{ a finite number of times.} \]

\[ [\pi^*](\varphi \rightarrow [\pi]\varphi) \rightarrow ([\varphi \rightarrow [\pi^*]\varphi) : \rightsquigarrow \text{Exercise.} \]

Do these formulae always hold?

How can we actually use this logic?
5 From Modal to Dynamic Logic

5.5 Dynamic Logic

But what if we want to consider complex programs? First of all, we have to make sure that we can build such programs.

**Definition 5.42 (Composite labels)**

We require that the set of labels forms a Kleene algebra \( \langle L, ;, \cup, \ast \rangle \). We also assume that the set of labels contains constructs of the form “\( \varphi \)?”, whenever \( \varphi \) is a formula not involving any modalities.

**Definition 5.43 (Condition on Labels)**

We assume that the labels obey the following conditions:

- \( s \xrightarrow{\alpha ; \beta} t \) iff \( s \xrightarrow{\alpha} s' \) and \( s' \xrightarrow{\beta} t \),
- \( s \xrightarrow{\alpha \cup \beta} t \) iff \( s \xrightarrow{\alpha} t \) or \( s \xrightarrow{\beta} t \),
- \( s \xrightarrow{\ast} t \) is the reflexive and transitive closure of \( s \xrightarrow{\alpha} t \),
- \( s \xrightarrow{\varphi} t \) iff \( s = t \) and \( s \models_M \varphi \).

What has this to do with programs?

- “;” means sequential composition,
- “\( \cup \)” means nondeterministic choice,
- “\( \ast \)” means finite iteration (regular expr.),
- “\( \varphi \)?” means test.

**if \( \varphi \) then \( a \) else \( b \) \( (\varphi; a) \cup (\neg \varphi; b) \)**

**while \( \varphi \) do \( a \) \( (\varphi; a)^\ast ; (\neg \varphi) \)**
We are now ready to define the semantics of DL for arbitrary complex expressions of labels.

**Definition 5.44 (Semantics of DL)**

We assume that the set of labels forms a *Kleene algebra* and that the conditions of Definition 5.43 hold. Then we define, as in Definition 5.41:

\[ M, s \models [\alpha] \phi \iff \text{for all } t \text{ such that } s \xrightarrow{\alpha} t, \text{ we have } M, t \models \phi. \]

**Example 5.45 (Transition System: Rocket and Cargo)**

We consider bringing some cargo with a rocket from one place to another. So we have a *rocket* and a *cargo*, that is moved between London (proposition \( r_{oL} \)) and Paris (proposition \( r_{oP} \)). The cargo can be in London (\( c_{aL} \)), Paris (\( c_{aP} \)), or inside the rocket (\( c_{aR} \)).

- The rocket can be moved only if it has its fuel tank full (\( f_{uelOK} \)),
- When it moves, it consumes fuel, and \( n_{ofuel} \) holds after each flight.

This determines the underlying propositional language.
5 From Modal to Dynamic Logic

5.5 Dynamic Logic

Prof. Dr. Jürgen Dix · Department of Informatics, TUC
Multiagent Systems II (MSc), SS 2012 553

5.6 Axiomatic systems for temporal logics

Weakly Completeness

Like many modal logics, LTL is only weakly complete, i.e.

\[ \Phi \models \psi \text{ implies } \Phi \vdash \psi \]

is only true for finite sets \( \Phi \), not for infinite sets.
The set

\[ \{ r \rightarrow s, r \rightarrow Xs, r \rightarrow XXs, \ldots \} \]

serves as a counterexample. It certainly implies \( r \rightarrow Gs \), but this can not be inferred using any sound axiom system (the reason is that no finite subset of the above set implies this formula).

Note that

- we have “¬”, “∨”, as basic propositional operators (all the others are macros), and
- “· U ·”, and “X ·” as basic LTL operators,
- All other operators are defined as usual.
The system consisting of Hilbert\(^{SL}\) and the following

\[
\begin{align*}
(A_1) & \quad G(\varphi \rightarrow X\varphi) \rightarrow (\varphi \rightarrow G\varphi) \\
(A_2) & \quad (\varphi \cup \psi) \leftrightarrow (\psi \cup (\varphi \land X(\varphi \psi))) \\
(A_3) & \quad (\varphi \cup \psi) \rightarrow F\psi \\
(Fun) & \quad \neg X\varphi \leftrightarrow X\neg\varphi \\
(K_x) & \quad X(\varphi \rightarrow \varphi') \rightarrow (X\varphi \rightarrow X\varphi') \\
(K_C) & \quad G(\varphi \rightarrow \varphi') \rightarrow G(G\varphi \rightarrow G\varphi') \\
(N_x) & \quad \neg X\varphi \\
(N_G) & \quad \neg G\varphi
\end{align*}
\]

is sound and weakly complete with respect to \(LTL\) Kripke models.

Theorem 5.47 (Axiomatization of CTL)

The system consisting of Hilbert\(^{SL}\) and the following

\[
\begin{align*}
(A_1) & \quad EF\varphi \leftrightarrow E(T \varphi) & (A_1') & \quad AF\varphi \leftrightarrow A(T \varphi) \\
(A_2) & \quad AG\varphi \leftrightarrow \neg EF\neg\varphi & (A_2') & \quad EG\varphi \leftrightarrow \neg AF\neg\varphi \\
(A_3) & \quad EX(\varphi \vee \psi) \leftrightarrow (EX\varphi \vee EX\psi) & (A_4) & \quad AX\varphi \leftrightarrow \neg EX\neg\varphi \\
(A_5) & \quad E \land AXT & (R) & \quad \varphi \rightarrow A\varphi \\
(A_6) & \quad E(\varphi \cup \psi) \leftrightarrow (\psi \vee (\varphi \land EX(\varphi \psi))) \\
(A_6') & \quad A(\varphi \cup \psi) \leftrightarrow (\psi \vee (\varphi \land AXA(\varphi \psi))) \\
(A_7) & \quad AG(\rho \rightarrow (\neg\psi \land EX\rho)) \rightarrow (\rho \rightarrow \neg A(\varphi \psi)) \\
(A_8) & \quad AG(\rho \rightarrow (\neg\psi \land EX\rho)) \rightarrow (\rho \rightarrow \neg AF\psi) \\
(A_9) & \quad AG(\rho \rightarrow (\neg\psi \land (\varphi \land AX\rho))) \rightarrow (\rho \rightarrow \neg E(\varphi \psi)) \\
(A_{10}) & \quad AG(\rho \rightarrow (\neg\psi \land AX\rho)) \rightarrow (\rho \rightarrow \neg EF\psi) \\
(A_{11}) & \quad AG(\varphi \rightarrow \psi) \rightarrow (EX\varphi \rightarrow EX\psi)
\end{align*}
\]

is sound and weakly complete with respect to \(CTL\) Kripke models.

Note that

- we have “\(\neg\)”, “\(\lor\)”, as basic propositional operators (all the others are macros), and
- “\(E \cdot \mathcal{U}.\)”, “\(\mathcal{E}.\)”, and “\(\mathcal{E}G.\)”, as basic \(CTL\) operators,
- All other operators are defined as usual.

A (very complicated) sound and complete (with respect to the appropriate Kripke models) axiomatization of \(CTL^*\) has been defined in \[?\].
5 From Modal to Dynamic Logic

5.6 Axiomatic systems for temporal logics

Note that
- we have “¬”, “∨”, as basic propositional operators (all the others are macros), and
- “⟨⟨A⟩⟩ X.”, “⟨⟨A⟩⟩ □.”, “⟨⟨A⟩⟩ · U.”, as basic ATL operators,
- all other operators are defined as usual, and
- we only consider the version of ATL based on perfect information and perfect recall: ATL\text{IR} (=ATL_{IR}).

Theorem 5.48 (Axiomatization of ATL)

The system consisting of Hilbert\text{SL} and the following (where \(A, A_1, A_2\) are subsets of \(\text{Agt}\) and \(A_1, A_2\) are disjoint):

\[
\begin{align*}
(\bot) & \quad \neg \langle\langle A\rangle\rangle X \bot \\
(T) & \quad \langle\langle A\rangle\rangle X \top \\
(A) & \quad \langle\langle 0\rangle\rangle X \neg \varphi \rightarrow \langle\langle \text{Agt}\rangle\rangle X \varphi \\
(S) & \quad (\langle\langle A_1\rangle\rangle X \varphi_1 \land \langle\langle A_2\rangle\rangle X \varphi_2) \rightarrow (\langle\langle A_1 \cup A_2\rangle\rangle X (\varphi_1 \land \varphi_2)) \\
(FP_\Box) & \quad (\langle\langle \Box\rangle\rangle \varphi) \leftrightarrow (\varphi \land \langle\langle \Box\rangle\rangle X \langle\langle \Box\rangle\rangle \varphi) \\
(GFP_\Box) & \quad (\langle\langle \emptyset\rangle\rangle \Box \theta \rightarrow (\varphi \land \langle\langle A\rangle\rangle X \theta)) \rightarrow (\langle\langle \emptyset\rangle\rangle \Box \theta \rightarrow \langle\langle A\rangle\rangle \Box \varphi) \\
(FP_U) & \quad (\langle\langle A\rangle\rangle \varphi_1 U \varphi_2 \leftrightarrow (\varphi_2 \lor (\varphi_1 \land \langle\langle A\rangle\rangle X \langle\langle A\rangle\rangle \varphi_1 U \varphi_2))) \\
(LFP_U) & \quad (\langle\langle \emptyset\rangle\rangle \Box (\varphi_2 \lor (\varphi_1 \land \langle\langle \emptyset\rangle\rangle X \theta)) \rightarrow \theta) \rightarrow (\langle\langle \emptyset\rangle\rangle \Box \varphi_1 U \varphi_2 \rightarrow \theta)
\end{align*}
\]

is sound and weakly complete with respect to ATL models (concurrent game structures).

This axiomatization is from [7]. Nothing is known for ATL\text{*}, ATL\text{+}, ATL\text{IR} or ATL\text{IR}.
6. Cooperative Agents

6.1 Alternating-Time Temporal Logics

The picture so far.

What kind of logics did we introduce so far?

- Linear-time temporal logic (LTL)
- Branching-time logics (CTL and CTL*)

In the temporal case each transition modelled a time step. We considered only one single “actor”.

Now: Modelling abilities of multiple agents: CTL can be viewed as the single actor restriction of ATL.

Agents can execute actions and cooperate. Action profiles determine the behaviour of the system.

Outline

- We introduce ATL, Alternating Time Temporal Logic: a blend of temporal logic and game theory.
- Like CTL, ATL comes in two variants: ATL and ATL*.
- Appropriate models for ATL are concurrent game structures.
- We introduce four variants of ATL along two different axis:
  - perfect vs imperfect information, and
  - perfect vs imperfect recall.
Alternating-time Temporal Logics

- **ATL, ATL⁺ [Alur et al. 1997]**
- **Temporal logic meets game theory**
- **Modeling abilities of multiple agents**
- **Main idea: cooperation modalities**

**⟨⟨A⟩⟩φ:** coalition A has a collective strategy to bring about φ

**Bringing about** is understood in the game-theoretical sense: There is a winning strategy.

The syntax is given as for the computation-tree logics.

**Definition 6.1 (Language L_ATL⁺ [Alur et al., 1997])**

The language \( L_{ATL⁺} \) is given by all formulae generated by the following grammar:

\[
\varphi ::= p \mid \neg \varphi \mid \varphi \lor \psi \mid \langle\langle A \rangle\rangle\gamma \quad \text{where} \\
\gamma ::= \varphi \mid \neg \gamma \mid \gamma \lor \gamma \mid \gamma U \gamma \lor \Box \gamma,
\]

\( A \subseteq \text{Agt} \) and \( p \in \text{Prop} \). Formulae \( \varphi \) (resp. \( \gamma \)) are called state (resp. path) formulae.

Note that we are using now the symbol “\( \Box \)” instead of “\( \Box \)” as it is more customary when dealing with ATL.

The language \( L_{ATL} \) restricts \( L_{ATL⁺} \) in the same way as \( L_{CTL} \) restricts \( L_{CTL⁺} \):

Each temporal operator must be directly preceded by a cooperation modality.

**Definition 6.2 (Language L_ATL [Alur et al., 1997])**

The language \( L_{ATL} \) is given by all formulae generated by the following grammar:

\[
\varphi ::= p \mid \neg \varphi \mid \varphi \lor \psi \mid \langle\langle A \rangle\rangle \Box \varphi \mid \langle\langle A \rangle\rangle \Box \varphi
\]

where \( A \subseteq \text{Agt} \) and \( p \in \text{Prop} \).

Note that we are using now the symbol “\( \Box \)” instead of “\( \Box \)” as it is more customary when dealing with ATL.

The language \( L_{ATL⁺} \) restricts \( L_{ATL⁺} \) but extends \( L_{ATL} \). It allows for Boolean combinations of path formulae.

**Definition 6.3 (Language L_ATL⁺)**

The language \( L_{ATL⁺} \) is given by all formulae generated by the following grammar:

\[
\varphi ::= p \mid \neg \varphi \mid \varphi \lor \psi \mid \langle\langle A \rangle\rangle \gamma \quad \gamma ::= \neg \gamma \mid \gamma \lor \gamma \mid \Box \varphi \mid \varphi U \varphi
\]

where \( A \subseteq \text{Agt} \) and \( p \in \text{Prop} \).
6 Cooperative Agents

6.1 Alternating-Time Temporal Logics

**ATL Models: Concurrent Game Structures**

- Agents, actions, transitions, atomic propositions
- Atomic propositions + interpretation
- Actions are abstract

**Definition 6.4 (Concurrent Game Structure)**

A *concurrent game structure* is a tuple $\mathcal{M} = \langle \text{Agt}, Q, \pi, \text{Act}, d, o \rangle$, where:

- $\text{Agt}$: a finite set of all *agents*;
- $Q$: a set of *states*;
- $\pi: Q \rightarrow 2^\text{Prop}$: a *valuation* of propositions;
- $\text{Act}$: a finite set of (atomic) *actions*;
- $d: \text{Agt} \times Q \rightarrow 2^\text{Act}$ defines actions available to an agent in a state;
- $o$: a deterministic *transition function* that assigns outcome states $q' = o(q, \alpha_1, \ldots, \alpha_k)$ to states and tuples of actions.

**Recall and information**

A *strategy* of agent $a$ is a *conditional plan* that specifies what $a$ is going to do in each situation.

Two types of “situations”: Decisions are based on

- the current state only ($\leftrightarrow$ *memoryless strategies*)
  
  $s_a: Q \rightarrow \text{Act}$.

- on the whole history of events that have happened ($\leftrightarrow$ *perfect recall strategies*)
  
  $s_a: Q^+ \rightarrow \text{Act}$. 

We also distinguish between agents with

- *perfect information* (all states are distinguishable).

- *imperfect information* (some state are indistinguishable).
**Perfect Information Strategies**

**Definition 6.5 (IR- and Ir-strategies)**

- A perfect information perfect recall strategy for agent \( a \) (IR-strategy for short) is a function
  \[
  s_a : Q^+ \to \text{Act} \quad \text{such that} \quad s_a(q_0q_1 \ldots q_n) \in d_a(q_n).
  \]
  The set of such strategies is denoted by \( \Sigma_a^{\text{IR}} \).

- A perfect information memoryless strategy for agent \( a \) (Ir-strategy for short) is given by a function
  \[
  s_a : Q \to \text{Act} \quad \text{where} \quad s_a(q) \in d_a(q).
  \]
  The set of such strategies is denoted by \( \Sigma_a^{\text{Ir}} \).

\( i \) (resp. \( I \)) stands for imperfect (resp. perfect) information and \( r \) (resp. \( R \)) for imperfect (resp. perfect) recall. [Schobbens, 2004]

**Outcome of a strategy**

\( \text{out}(q, s_A) = \text{set of all paths that may occur} \)
when agents \( A \) execute \( s_A \) from state \( q \) onward.

**Definition 6.6 (Outcome)**

\[
\lambda = q_0q_1 \ldots \in Q \in \text{out}(q, s_A) \subseteq Q^* \quad \text{iff} \]

1. \( q_0 = q \)
2. For each \( i = 1, \ldots \) there is a tuple \((\alpha_1^{-1}, \ldots, \alpha_k^{-1}) \in \text{Act}^k\) such that
   - \( \alpha_a^{-1} \in d_a(q_i) \) for each \( a \in \text{Agt} \),
   - \( \alpha_a^{-1} = s_A(q_0q_1 \ldots q_i) \) for each \( a \in A \), and
   - \( s(q_{i-1}, \alpha_1^{-1}, \ldots, \alpha_k^{-1}) = q_i \).

For an Ir-strategy replace “\( s_A(q_0q_1 \ldots q_{i-1}) \)” by “\( s_A(q_{i-1}) \)”.

**Some Notation**

The following holds for all kind of strategies:

- A collective strategy for a group of agents \( A = \{a_1, \ldots, a_r\} \subseteq \text{Agt} \) is a set
  \[
  s_A = \{s_a \mid a \in A\}
  \]
  of strategies, one per agent from \( A \).
- \( s_A|_a \) we denote agent \( a \)'s part of the collective strategy \( s_A \), \( s_A|_a = s_A \cap \Sigma_a \).
- \( s_0 = \emptyset \) denotes the strategy of the empty coalition.
- \( \Sigma_A \) denotes the set of all collective strategies of \( A \).
- \( \Sigma = \Sigma_{\text{Agt}} \)

**Definition 6.7 (Perfect information semantics)**

- \( M, q \models_{\text{I}} p \) iff \( p \) is in \( \pi(q) \);
- \( M, q \models_{\text{I}} \varphi \lor \psi \) iff \( M, q \models_{\text{I}} \varphi \) or \( M, q \models_{\text{I}} \psi \);
- \( M, q \models_{\text{I}} (A) \Phi \) iff there is a collective Ix-strategy \( s_A \) such that,
  for each path \( \lambda \in \text{out}(q, s_A) \), we have
  \( M, \lambda \models_{\text{I}} \Phi \).
- \( M, \lambda \models_{\text{I}} \lnot \varphi \) iff \( M, \lambda[1, \infty] \models_{\text{I}} \varphi \);
- \( M, \lambda \models_{\text{I}} \varphi[i, \infty] \) iff \( M, \lambda[i, \infty] \models_{\text{I}} \varphi \) for some \( i \geq 0 \);
- \( M, \lambda \models_{\text{I}} \varphi \) iff \( M, \lambda[i, \infty] \models_{\text{I}} \varphi \) for all \( i \geq 0 \);
- \( M, \lambda \models_{\text{I}} \varphi[U \psi] \) iff \( M, \lambda[i, \infty] \models_{\text{I}} \psi \) for some \( i \geq 0 \), and
  \( M, \lambda[j, \infty] \models_{\text{I}} \varphi \) for all \( 0 \leq j \leq i \).

Note that temporal formulae and the Boolean connectives are handled as before.
6 Cooperative Agents

6.1 Alternating-Time Temporal Logics

Example: Robots and Carriage

\[ \text{Example: Robots and Carriage} \]

\[ q_0 \xrightarrow{\text{wait,wait}} q_1 \xrightarrow{\text{push,wait}} q_0 \]

\[ \text{Definition 6.8 (ATL}_{ix}, ATL_{ix}^+, ATL_{ix}^*, ATL, ATL^*) \]

We define ATL_{ix}, ATL_{ix}^+, and ATL_{ix}^* as the logics \((L_{ATL_{ix}}, |_{Ix})\), \((L_{ATL_{ix}^+}, |_{Ix})\) and \((L_{ATL_{ix}^*}, |_{Ix})\) where \(x \in \{r, R\}\), respectively. Moreover, we use ATL (resp. ATL^*) as an abbreviation for ATL_{IR} (resp. ATL_{IR}^*).

Intuitively, a logic is given by the set of all valid formulae.

6.2 Perfect vs.imperfect recall

\[ \text{Theorem 6.9} \]

For \(L_{ATL}\), the perfect recall semantics is equivalent to the memoryless semantics under perfect information, i.e.,

\[ M, q \models_{IR} \varphi \iff M, q \models_{IR} \varphi. \]

Both semantics are different for \(L_{ATL^*}\). That is

\[ \text{ATL} = \text{ATL}_{IR} = \text{ATL}_{IR}^*. \]

\[ \text{Proof idea.} \]

The first “non-looping part” of each path has to satisfy a formula.

Exercise

The property has been first observed in [Schobbens, 2004] but it follows from [Alur et al. 2002] in a straightforward way.
### Example: Robots and Carriage (2)

![Diagram of Robots and Carriage]

What about $\langle\langle 1, 2 \rangle\rangle (\Diamond \text{pos}_1 \wedge \Diamond \text{halt})$?

- $\mathcal{M}, q_0 \models \text{IR} \langle\langle 1, 2 \rangle\rangle (\Diamond \text{pos}_1 \wedge \Diamond \text{halt})$
- $\mathcal{M}, q_0 \not\models \text{Ir} \langle\langle 1, 2 \rangle\rangle (\Diamond \text{pos}_1 \wedge \Diamond \text{halt})$

---

### IR-Tree Unfolding

- Interesting is the comparison between memory and no memory.
- Can Agents really achieve more (in terms of validities) if they have memory available?
- Suppose we want to show that $\text{ATL}_\text{Ir}^* \subseteq \text{ATL}_\text{IR}^*$; i.e., more properties of games are valid if perfect recall strategies are considered.
- For this purpose, we show that every $\text{IR}$-satisfiable formula is also $\text{Ir}$-satisfiable.
- Then, the claim follows: Suppose $\varphi \in \text{ATL}_\text{Ir}$ and $\varphi \notin \text{ATL}_\text{IR}$. By the latter, $\neg \varphi$ is $\text{IR}$-satisfiable hence also $\text{Ir}$-satisfiable. Contradiction!

How can we show that IR-satisfiability implies Ir-satisfiability?

---

### Definition 6.11 (Tree-like CGS)

Let $\mathcal{M}$ be a CGS. $\mathcal{M}$ is called tree-like iff there is a state $q_0$ (the root) such that for every $q$ there is a unique history leading from $q_0$ to $q$.

### Proposition 6.12 (Recall invariance for tree-like CGS)

For every tree-like CGS $\mathcal{M}$, state $q$ in $\mathcal{M}$, and $\text{ATL}^*$-formula $\varphi$, we have: $\mathcal{M}, q \models \varphi$ iff $\mathcal{M}, q \models \text{Ir} \varphi$.

Can we always obtain such a tree-like “version” of a model?
For each model, we can construct an equivalent tree-like model: Fix a state and unfold the model to an infinite tree.

Definition 6.13 (Perfect information tree unfolding)

Let $M = (A, Q, II, \pi, Act, d, o)$ be a CGS and $q$ be a state in it. The (perfect information) tree unfolding of the pointed model $(M, q)$ denoted $T(M, q)$ is defined as $(A, Q', Prop, \pi', Act, d', o')$ where

- $Q' := \Lambda_{fin}^M(q)$,
- $d'(a, h) := d(a, last(h))$,
- $o'(h, \vec{\alpha}) := h \circ o(last(h), \vec{\alpha})$, and
- $\pi'(h) := \pi(last(h))$.

The node $q$ in the unfolding is called root of $T(M, q)$.

Theorem 6.14

For every CGS $M$, state $q$ in $M$, and ATL$^*$-formula $\varphi$ we have:

$M, q \models IR \varphi \iff T(M, q), q \models IR \varphi \iff T(M, q), q \models IR \varphi$.

We now compare perfect vs. imperfect memory.

Proposition 6.15

$ATL^*_{IR} \subseteq ATL^*_{IR}$ (In fact: $ATL^*_{IR} \subseteq ATL^+_{IR}$)

Membership: If $\models_{IR} \varphi$ then Treemodels $\models_{IR} \varphi$ then

Treemodels $\models_{IR} \varphi$ then $\models_{IR} \varphi$

Strict inclusion:

$M, q_0 \not\models_{IR} \langle \langle a \rangle \rangle(p_1 \land p_2) \leftrightarrow \langle \langle a \rangle \rangle((p_1 \land \langle \langle a \rangle \rangle \land p_2) \lor (p_2 \land \langle \langle a \rangle \rangle \land p_1))$.

$p_1 = \text{clean} $

$p_2 = \text{delivered}$

6.3 Imperfect Information
Imperfect information

How can we reason about agents/extensive games with imperfect information?

We combine ATL* and epistemic logic.

- We extend CGSs with indistinguishability relations $\sim_a \subseteq Q \times Q$, one per agent. The relations are assumed to be equivalence relations.
- We interpret $\langle\langle A \rangle\rangle$ epistemically ($\models_{iR}$ and $\models_{ir}$)

Definition 6.16 (CEGS)

A concurrent epistemic game structure (CEGS) is a tuple

$\mathcal{M} = (\text{Agt}, Q, \Pi, \text{Act}, d, o, \{\sim_a \mid a \in \text{Agt}\})$

with

- $(\text{Agt}, Q, \Pi, \text{Act}, d, o)$ a CGS and
- $\sim_a \subseteq Q \times Q$ equivalence relations (indistinguishability relations).

Example: Robots and Carriage

The last example shows that although there is a strategy, the agents do not know it (because of imperfect information).

Problem:

Strategic and epistemic abilities are not independent!

$\langle\langle \text{Agt} \rangle\rangle \Phi = A$ can bring about $\Phi$

It should at least mean that $A$ are able to identify and execute the right strategy!

Executable strategies = uniform strategies
**Definition 6.17 (Uniform strategy)**

Strategy $s_a$ is uniform iff it specifies the same choices for indistinguishable situations:
- Memoryless strategies:
  \[ \text{if } q \sim_a q' \text{ then } s_a(q) = s_a(q'). \]
- Perfect recall:
  \[ \text{if } \lambda \approx_a \lambda' \text{ then } s_a(\lambda) = s_a(\lambda'), \]
  where $\lambda \approx_a \lambda'$ iff $\lambda[i] \sim_a \lambda'[i]$ for every $i$.

A collective strategy is uniform iff it consists only of uniform individual strategies.

**Imperfect Information Strategies**

**Definition 6.18 (IR- and Ir-strategies)**

A imperfect information perfect recall strategy for agent $a$ (IR-strategy for short) is a uniform IR-strategy.

A imperfect information memoryless strategy for agent $a$ (Ir-strategy for short) is a uniform Ir-strategy.

The outcome is defined as before.

**Imperfect Information Semantics**

The imperfect information semantics is defined as before, only the clause for

\[ M, q \models \text{Ix}[\langle A \rangle] \varphi \text{ iff there is a collective Ix-strategy } s_A \text{ such that, for each path } \lambda \in \text{out}(q, s_A), \text{ we have } M, \lambda \models \text{Ix } \varphi. \]

is replaced by ($x \in \{r, R\}$ and $\sim_A := \bigcup_{a \in A} \sim_a$.)

\[ M, q \models \text{Ix}[\langle A \rangle] \varphi \text{ iff there is a uniform Ix-strategy } s_A \text{ such that, for each path } \lambda \in \bigcup_{q' : q \sim_A q'} \text{out}(q', s_A), \text{ we have } M, \lambda \models \text{Ix } \varphi. \]

**Remark 6.19**

This definition models that “everybody in $A$ knows that $\varphi$.”

The fixed-point characterization does not hold anymore!

**Theorem 6.20**

The following formulae are not valid for ATL$_{ir}$:

- $\langle A \rangle \Box \varphi \iff \varphi \land \langle A \rangle \bigcirc \langle A \rangle \Box \varphi$
- $\langle A \rangle \varphi_1 U \varphi_2 \iff \varphi_2 \lor (\varphi_1 \land \langle A \rangle \bigcirc \langle A \rangle \varphi_1 \varphi_2).

Proof: $\sim$ Exercise.
6 Cooperative Agents
6.3 Imperfect Information

Proof idea

We construct a counterexample for

\[ \langle\langle 1\rangle\rangle \Diamond p \leftrightarrow p \lor \langle\langle 1\rangle\rangle \circ \langle\langle 1\rangle\rangle \Diamond p \]

\[ M, q_1 \not\models_{ir} \langle\langle 1\rangle\rangle \Diamond p \text{ iff } \neg (\exists s \in \Sigma_a \forall \lambda \in \bigcup_{q \in \{q_1, q_2\}} \text{out}(q, s) \exists i \in \mathbb{N}_0 : M, \lambda[i] \models_{ir} p) \]

\[ M, q_1 \models_{ir} p \lor \langle\langle 1\rangle\rangle \circ \langle\langle 1\rangle\rangle \Diamond p \]

Proposition 6.21

**Comparing ATL\textsubscript{ir} vs. ATL\textsubscript{Ir}**

Incomplete vs. perfect information.

**Proposition 6.22**

\[ \text{ATL}_{ir} \subsetneq \text{ATL}_{Ir} \]

\[ M, q_1 \not\models_{ir} \langle\langle a\rangle\rangle \Diamond \text{shot} \rightarrow (\text{shot} \lor \langle\langle a\rangle\rangle \circ \langle\langle a\rangle\rangle \Diamond \text{shot}) \]

Proposition 6.22

**\text{ATL}_{ir} \subsetneq \text{ATL}_{Ir}**

\[ M, q_1 \not\models_{ir} \langle\langle a\rangle\rangle \Diamond \text{shot} \rightarrow (\text{shot} \lor \langle\langle a\rangle\rangle \circ \langle\langle a\rangle\rangle \Diamond \text{shot}) \]

\[ M, q_1 \models_{ir} p \lor \langle\langle 1\rangle\rangle \circ \langle\langle 1\rangle\rangle \Diamond p \]

**iR-Tree Unfolding**

The tree unfolding for the \textit{i}-semantics is more sophisticated. Consider the following model and the formula \(\langle\langle a\rangle\rangle \circ \langle\langle a\rangle\rangle \circ \langle\langle a\rangle\rangle \circ \text{shot}\). How can an \textit{iR}-tree unfolding look like?
A first approach is to connect separate unfoldings of the indistinguishable states by epistemic links.

Figure 31: Two separate unfoldings connected by an epistemic link. We use number $i_1 i_2 \ldots$ to refer to the history $q_1, q_1, \ldots$.

What about the formula $\langle\langle a \rangle\rangle \circ \langle\langle a \rangle\rangle \circ \langle\langle a \rangle\rangle \circ \text{shot}$? The $iR$-tree unfoldings is shown on the next slide.

Now we can state our main result for $iR$-tree unfoldings.

Theorem 6.23

For every CEGS $M$, state $q$ in $M$, and ATL* formula $\varphi$, it holds that

$M, q \models_{\text{IR}} \varphi$ iff $T_s(M, q), q \models_{\text{IR}} \varphi$ iff $T_s(M, q), q \models_{\text{ir}} \varphi$.

Summary

If a formula is $IR$-, or $iR$-satisfiable then it also is $IR$-, or $ir$-satisfiable, respectively.

Remark 6.24 (Important Validities and Invalidities)

- $\langle\langle a \rangle\rangle \diamond p \leftrightarrow p \lor \langle\langle a \rangle\rangle \circ \langle\langle a \rangle\rangle \diamond p$
  - Invalid in all variants with imperfect information.
  - Valid for perfect information.

- $\langle\langle a \rangle\rangle \langle\langle p_1 \land p_2 \rangle\rangle \leftrightarrow \langle\langle a \rangle\rangle \langle\langle (p_1 \land \langle\langle a \rangle\rangle \diamond p_2) \lor (p_2 \land \langle\langle a \rangle\rangle \diamond p_1) \rangle\rangle$
  - Invalid for imperfect information
  - Valid for perfect information and perfect recall

- $\langle\langle \emptyset \rangle\rangle \diamond \neg p \leftrightarrow \langle\langle \text{Agt} \rangle\rangle \square p$
  - Invalid for imperfect information
  - Valid for perfect information.
Overview of the Results

- “All” semantic variants are different on the level of general properties; before our study, it was by no means obvious.
- Strong pattern of subsumption (memory and information)
- Very natural (not obvious before).
- Non-validities are interesting.

Aim

We would like to

... reason about rational behavior of agents.

... have a logic that can use & describe solution concepts.

... compare different game theoretic solution concepts.

For this section we refer to [Bulling et al., 2008] for further details.
The Base Logic: $L_{\text{base}}^{\text{ATLP}}$

**Definition 6.25 ($L_{\text{base}}^{\text{ATLP}}$)**

The language $L_{\text{base}}^{\text{ATLP}}$ is defined over nonempty sets:

- $\text{Prop}$ of propositions, $p \in \text{Prop}$,
- $\text{Agt}$ of agents, $a \in \text{Agt}$, $A \subseteq \text{Agt}$, and
- $\Omega$ of basic plausibility terms, $\omega \in \Omega$.

$$\varphi ::= p \mid \neg \varphi \mid \varphi \land \varphi \mid \langle\langle A \rangle\rangle \square \varphi \mid \langle\langle A \rangle\rangle \varphi \mathcal{U} \varphi \mid \text{Pl}_A \varphi \mid (\text{set-pl } \omega) \varphi$$

**Semantics**

$\text{Pl}_B :$ Assuming plausible play of $B$

$$M, q \models \text{Pl}_B \langle\langle A \rangle\rangle \gamma$$

is true iff

1. $A$ can enforce $\gamma$ if
2. agents in $B$ play only plausible strategies

Which strategies are plausible?

**Plausibility Terms**

$\Omega$: Set of basic plausibility terms, $\omega \in \Omega$

- **Hard-wired** sets of strategies:
  - $\omega_{\text{NE}}$: Nash equilibria
  - $\omega_{\text{PO}}$: Pareto optimal strategies

$\text{(set-pl } \omega) :$ Sets plausible strategies to $[\omega] \subseteq \Sigma$

**Concurrent game structures with plausibility**

$$M = (\text{Agt}, Q, \text{Prop}, \pi, \text{Act}, d, \delta, \Upsilon, \Omega, [\cdot])$$

- $\Upsilon \subseteq \Sigma$: set of (plausible) strategy profiles
  - Example: $\Upsilon = \{(\text{head}, \text{head})\}$

- $\Omega = \{\omega_1, \omega_2, \ldots\}$: set of plausibility terms
  - Example: $\omega_{\text{NE}}$ stands for all Nash equilibria

- $[\cdot] : Q \rightarrow (\Omega \rightarrow \mathcal{P}(\Sigma))$: plausibility mapping, it assigns a set of strategy profiles to each state and plausibility term
  - Example: $[\omega_{\text{NE}}]_q = \{(\text{head}, \text{head}), (\text{tail}, \text{tail})\}$

And where do the terms come from?
Semantics of $\mathcal{L}_{ATLP}$

Let $P \subseteq \Sigma$ be a set of strategy profiles.

$\Sigma_A(P)$: strategy profiles of $A$ that are consistent with $P$.

Restricting $A$’s strategies wrt $P$

$\Sigma_A(P) := \{ s_A \in \Sigma_A \mid \exists t \in P \ (t[A] = s_A) \}$

$P(s_A)$: plausible strategy profiles of $A$ that agree on $s_A$.

Restricting $A$’s opponents strategies wrt $P$

$P(s_A) := \{ t \in P \mid t[A] = s_A \}$

$t[A]$: restriction of $t \in \Sigma$ to the strategy profile of $A$.

Outcome of a strategy

Outcome = Paths that may occur when agents $A$ perform $s_A$

$out(q, s_A, P) = \{ q_0q_1 \ldots \in Q^+ \mid q = q_0 \land \exists t \in P(s_A) \ \forall i \in \mathbb{N} \ (q_{i+1} = \delta(q_i, t(q_i))) \}$

The outcome is given wrt to a set of (plausible) strategy profiles $P$, restricting the opponents choices!

Semantics of $\mathcal{L}_{ATLP}$

We use a satisfaction relation $\models_P$ annotated with a set of strategy profiles.

$P$: strategies currently considered available

$m, q \models_P \langle A \rangle \bigcirc \varphi$ iff $\exists s_A \in \Sigma_A(P)$ $\forall \lambda \in \text{out}(q, s_A, P)$ $m, \lambda[1] \models_P \varphi$

$m, q \models_P \text{Pl} \varphi$ iff $m, q \models_\Sigma \varphi$

$m, q \models_P \text{Ph} \varphi$ iff $m, q \models_\Sigma \varphi$

$m, q \models_P (\text{set-pl } \omega) \varphi$ iff $m^\omega, q \models_\Sigma \varphi$ where the new model $m^\omega$ is equal to $m$ but the “new” set $\Upsilon^\omega$ of plausible strategy profiles is set to $\llbracket \omega \rrbracket$. 
6 Cooperative Agents
6.4 Reasoning About Rational Agents

Example: A Penny Game

Plausibility terms: abstract labels, no structure!
Idea: Formulas that describe plausible strategies!
Select all \( s \) \( s \) is better than any other strategy \( s' \)
Complex plausibility terms \( \omega \):
\[ \sigma \cdot \forall \sigma_1 \exists \sigma_2 \ldots \forall \sigma_n \{ \varphi(\sigma, \sigma_1, \ldots, \sigma_n) \} \]

Example: \( \omega_{\text{DOM}} = \sigma \cdot \forall \sigma' (\sigma \text{ better than } \sigma') \)

How to determine whether a strategy is good?

CGSP +preferences \( \leadsto \) Normal form game
Each CGSP with preferences corresponds to a normal form game.

No payoffs needed as for classical solution concepts!
Characterizing Solution Concepts

\[ BR_n^a(\sigma) \equiv (\text{set-pl } [\sigma' \mid_a]) \text{Pl}(\langle\alpha\rangle\eta_a \rightarrow (\text{set-pl } \sigma)\langle\emptyset\rangle\eta_a) \]

\[ NE_n^a(\sigma) \equiv \bigwedge_{a \in \text{Agt}} BR_n^a(\sigma) \]

\[ SPN_n^a(\sigma) \equiv \langle\emptyset\rangle \Box NE_n^a(\sigma) \]

\[ PO_n^a(\sigma) \equiv \exists \sigma' \text{ Pl}( \bigwedge_{a \in \text{Agt}} ((\text{set-pl } \sigma')\langle\emptyset\rangle\eta_a \rightarrow (\text{set-pl } \sigma)\langle\emptyset\rangle\eta_a) ) \lor \\
\forall_{a \in \text{Agt}} ((\text{set-pl } \sigma)\langle\emptyset\rangle\eta_a \land \neg (\text{set-pl } \sigma')\langle\emptyset\rangle\eta_a) \]
6.5 References


7. Programming Languages for MAS

- The Frame Problem
- (Con-)Golog
- (Concurrent-) MetaTeM
- Comparison
- References

Outline

We built on the situation calculus for first-order logic.
- We first show how to handle the frame axioms.
- Then we introduce Con-Golog, a programming language based on first-order dynamic logic and the situation calculus.
- Finally, MetaTeM is an executable specification language based on a temporal logic.

We end this chapter with a simple application and a comparison of how it is handled by the two frameworks we have introduced.

Symbolic AI: Symbolic representation, e.g. sentential or first order logic. Using deduction. Agent as a theorem prover.

Traditional: Theory about agents. Implementation as stepwise process (Software Engineering) over many abstractions.

Symbolic AI: View the theory itself as executable specification.
Internal state: Knowledge Base (KB), often simply called D (database).
A Solution to the Frame Problem?

Successor State Axioms
Where do the successor state axioms come from?

- We have to ask: Which fluents stay invariant?

  relational fluent:
  \[ \neg \text{broken}(x,s) \land (x \neq y \lor \neg \text{fragile}(x,s)) \rightarrow \neg \text{broken}(x,\text{do}(\text{drop}(r,y),s)) \]

  functional fluent:
  \[ \text{color}(x,s) = c \rightarrow \text{color}(x,\text{do}(\text{drop}(r,y),s)) = c \]

How many of such axioms do we need?

Number of Axioms
We need exactly \(2 \times \#\text{actions} \times \#\text{fluents}\).

Suppose we are given axioms of the form

\[ \ldots \rightarrow \text{fluent}(x,\text{do}(\text{action},s)) \]
\[ \ldots \rightarrow \neg\text{fluent}(x,\text{do}(\text{action},s)) \]

How can we compute the successor state axioms automatically?

Note, that the above set assumes implicitly that all actions can be applied: This is an overly optimistic assumption according to the Qualification Problem.

Qualification Problem Revisited
We assume a predicate Poss to describe the possibility to apply an action.

\[ \text{Poss}(\text{pickup}(r,x),s) \rightarrow \forall z (\neg \text{holding}(r,z,s)) \]

But \(\rightarrow\) is too weak. Can we replace it by \(\leftrightarrow\)?

- Actions have preconditions: necessary and sufficient conditions which must hold before an action can be executed.

  A robot \(r\) can lift object \(x\) in situation \(s\) iff \(r\) is not holding anything, is located next to \(x\), and \(x\) is not too heavy.
The effect on Broken when r drops an object (positive effect axiom):
\[ \text{Poss}(\text{drop}(r, x), s) \land \text{Fragile}(x) \rightarrow \text{Broken}(x, \text{do}\{\text{drop}(r, x), s\}). \]

Repairing an object causes the object not to be broken any more (negative effect axiom):
\[ \text{Poss}(\text{repair}(r, x), s) \rightarrow \neg \text{Broken}(x, \text{do}\{\text{repair}(r, x), s\}). \]

What about \( \text{Poss}(\text{pickup}(r, x), s) \rightarrow \forall z \ (\neg \text{holding}(r, z, s) \land \neg \text{heavy}(x) \land \text{nextto}(r, x, s)). \)

We suppose we are given a list of axioms of the form
\[ \text{Poss}(\text{action}(\vec{x}), s) \iff \phi_{\text{action}}(\vec{x}, s) \]
where \( \phi_{\text{action}}(\vec{x}, s) \) does not contain any do-terms.

Under which conditions could \( \neg \text{broken}(x, s) \) and \( \text{broken}(x, \text{do}(a, s)) \) be both true?
\[ (1'') : \neg \text{broken}(x, s) \land \text{broken}(x, \text{do}(a, s)) \rightarrow \exists r \ (a \doteq \text{drop}(r, x) \land \text{Fragile}(x) \lor \exists b \ (a \doteq \text{explode}(b) \land \text{nextto}(b, x, s)) \)
\]
\[ (2') : \text{broken}(x, s) \land \neg \text{broken}(x, \text{do}(a, s)) \rightarrow \exists r \ a \doteq \text{repair}(r, x) \]

We assume these are all possibilities for \( \text{broken}, \neg \text{broken} \). Then (1), (1') are equivalent to
\[ \exists r \ (a \doteq \text{drop}(r, x) \land \text{Fragile}(x)) \lor \exists b \ (a \doteq \text{explode}(b) \land \text{nextto}(b, x, s)) \]
\[ \rightarrow \text{broken}(x, \text{do}(a, s)). \]

(2) is equivalent to
\[ \exists r \ a \doteq \text{repair}(r, x) \rightarrow \neg \text{broken}(x, \text{do}(a, s)) \]
(1), (1'), (2), (1'"), (2') are equivalent to the successor state axiom

\[
\text{broken}(x, \text{do}(a, s)) \iff \\
\exists r \ (a = \text{drop}(r, x) \land \text{fragile}(x, s)) \lor \\
\exists a \ (b = \text{explode}(b) \land \text{nextto}(b, x, s)) \lor \\
\text{broken}(x, s) \land \neg \exists r \ a = \text{repair}(r, x)
\]

This is easily generalised to functional fluents!

Thus the \(2 \times \#\text{actions} \times \#\text{fluents}\) many axioms can be rewritten into only \#\text{fluents} many axioms (\(2 \times \#\text{fluents}\) if we count each equivalence twice). But we also need the \text{Poss} axioms: another \#\text{actions} many.

 Altogether, the \(2 \times \#\text{actions} \times \#\text{fluents}\) are compiled into (modulo a constant factor) \#\text{actions} + \#\text{fluents}.

Some people call this a solution to the frame problem.

---

7.2 (Con-)Golog

From now on we use the symbol

- \(\supset\) to denote implication \(\rightarrow\) and
- \(\equiv\) to denote equivalence \(\leftrightarrow\).

Firstly we have to give a precise definition as to how the domain we are dealing with is described: the effects of actions, how to denote and distinguish actions, the initial state.
Definition 7.1 (Domain Descriptions (Sit. Calc.))

A domain description $D$ in the situation calculus consists of the following: $D = D_{ap} \cup D_{ss} \cup D_{una} \cup D_{S0}$

- $D_{ap}$ = action preconditions
  $\text{Poss}(A(x), s) \equiv \Pi_A(x, s)$

- $D_{ss}$ = successor state axioms
  $\text{Poss}(a, s) \supset [F(x, \text{do}(a, s)) \equiv \Phi_F(x, a, s)]$

- $D_{una}$ = unique names for actions:
  $A(x) \neq B(y)$ for distinct $A$ and $B$, $A(u) = A(v) \supset u = v$.

- $D_{S0}$ = sentences describing the initial state, where $S_0$ is the only situation term (no do(_, _)).

---

Planning in the Situation Calculus

- Let $G(s)$ be a goal. Then we are interested in:
  $D \models \exists s G(s)$, and leave this task to a theorem prover.

- As a side effect of the derivation we can (often) read off the correct action sequence (a plan) from the term bound by $s$
  $s = \text{do}(a_n, \text{do}(a_{n-1}, \ldots \text{do}(a_1, S_0) \ldots))$.

  **Not quite right:** this allows action sequences that are not executable ($G$ a tautology).

---

Definition 7.2 (Plan)

1. We define a predicate $\text{Legal}(s)$ which is true if every action in the action sequence leading to $s$ is executable:
   $\text{Legal}(s) \equiv s = S_0 \lor \exists a \exists s'[s = \text{do}(a, s') \land \text{Poss}(a, s') \land \text{Legal}(s')]$

2. A plan is a sequence of ground action terms
   $a, a^{-1}, b, b^{-1}, \ldots$
   $\text{do}(a_n, \text{do}(a_{n-1}, \ldots \text{do}(a_1, S_0) \ldots))$
Question

How to find a suitable $s$?

First approach: **Regression planning**, that is planning backwards from the goal to the initial state.

Second approach: **Golog**, a programming language for the situation calculus.

We define a regression operator $\mathcal{R}$ which transforms a formula of the situation calculus and reduces the nesting of $\text{do}(\alpha, \text{do}(\ldots))$ terms by 1.

**Definition 7.3 (Regression operator $\mathcal{R}$)**

1. If $A$ is not a fluent (incl. $=\sim$) or $A = \text{Poss}(a, s)$, then $\mathcal{R}[A] = A$.
2. If $F$ is a fluent and $s$ a situation variable or $s = S_0$, then $\mathcal{R}[F(t, s)] = F(t, s)$.
3. If $F$ is a fluent with successor state axiom
   $$\text{Poss}(a, s) \supset [F(x_1, \ldots, x_n, \text{do}(a, s)) \equiv F[\bar{x}_1, \ldots, x_n, a, s]],$$
   then $\mathcal{R}[F(t_1, \ldots, t_n, \text{do}(\alpha, \sigma))] = F[\mathcal{R}[t_1], \ldots, \mathcal{R}[t_n], \mathcal{R}[\alpha], \mathcal{R}[\sigma]]$.

**Example 7.4**

Let $P, Q$ be fluents, $R$ not a fluent. Let the successor state axioms for $P$ and $Q$ be

$$\text{Poss}(a, s) \supset [P(x, s) \equiv \Phi_P(x, a, s)],$$

$$\text{Poss}(a, s) \supset [Q(\text{do}(a, s)) \equiv \Phi_Q(a, s)].$$

Then (neither $\Phi_P$ nor $\Phi_Q$ contain $\text{do}(\_, \_)$ terms):

$$\mathcal{R}[G] = \forall a \forall s[\Phi_P(A, a, \text{do}(a', s)) \land s \Rightarrow \text{do}(B, S_0) \supset \exists x[P(x, s) \land \text{Poss}(B, \text{do}(a, s)) \land R(x) \land Q(\text{do}(B, s))]].$$
Linear Regression

Let \( \alpha = A(g_1, \ldots, g_k) \) be a ground action with precondition axiom
\[
\text{Poss}(A(x_1, \ldots, x_k), s) \equiv \Pi_A(x_1, \ldots, x_k, s).
\]

Let \( \text{precond}(\alpha, s) \) be \( \Pi_A(g_1, \ldots, g_k, s) \). Let \( \phi(s) \) be a formula whose only situation term is the free variable \( s \). (\( \phi(s) \) may contain other free variables, but not of type situation.)

The regression of \( \phi(s) \) through \( \alpha \), denoted as \( R(\alpha, \phi(s)) \), is
\[
R[\phi(\text{do}(\alpha, s))] \land \text{precond}(\alpha, s).
\]

Theorem 7.5 (Reiter)

Let \( \alpha_1, \ldots, \alpha_n \) be a sequence of ground actions and \( G(s) \) a wff whose only free variable is \( s \). Then \( \alpha_1, \ldots, \alpha_n \) is a plan for \( G \) iff
\[
\mathcal{D}_{una} \cup \mathcal{D}_{S_0} \models R(\alpha_1, R(\alpha_2, \ldots R(\alpha_{n-1}, R(\alpha_n, G(s)))) \ldots))|_{S_0}.
\]

Linear Regression Planning

**Input:** a domain description \( \mathcal{D} \) and goal \( G(s) \).

**Output:** an action sequence \( A \).

Initialize \( A \) (empty sequence) and \( \Gamma \ (G(s)) \).

Check whether \( \mathcal{D}_{una} \cup \mathcal{D}_{S_0} \models \Gamma|_{S_0} \). If yes, then return \( A \).

Otherwise

- choose a ground action \( \alpha \);
- insert it at the beginning of \( A \);
- compute the regression of \( \Gamma \) through \( \alpha \);
- assign the result to \( \Gamma \);
- goto 2.

Comments on Regression Planning (1)

- Because of the previous theorem, the method is both correct and complete.
- The algorithm is nondeterministic. It does not say which \( \alpha \) to choose next. (Can be implemented using backtracking.)
Comments on Regression Planning (2)

- Does not always terminate, for example, if there is no plan. The problem whether a plan exists is undecidable.
- The algorithm is fairly dumb. For example, if there is an action `noop` with no effects, then this could be chosen arbitrarily often, i.e. plans need not be optimal.
- A simple strategy for goal-oriented search: try those actions which satisfy one or more conditions which do not already follow from the initial situation.

These result in successor state axioms $D_{ss}$:

- $\text{Poss}(a, s) \supset [\text{Holding}(x, \text{do}(a, s)) \equiv a = \text{pickup}(x) \lor \text{Holding}(x, s) \land a \neq \text{drop}(x)]$
- $\text{Poss}(a, s) \supset [\text{Broken}(x, \text{do}(a, s)) \equiv a = \text{drop}(x) \lor \text{Broken}(x, s)]$
- $\text{Poss}(a, s) \supset [\text{At}(x, \text{do}(a, s)) \equiv a = \text{walk}(x) \lor \text{At}(x, s) \land \neg \exists y (y \neq x \land a = \text{walk}(y))$

Initial situation $D_{S_0}$:

$\forall x \neg \text{Holding}(x, S_0), \neg \text{At}(A, S_0), \neg \text{Broken}(A, S_0)$

Goal: $\exists s \text{ Broken}(A, s)$ (Break A.)

Example 7.6

Actions: $\text{pickup}(x), \text{drop}(x), \text{walk}(x)$

Preconditions $D_{ap}$:

- $\text{Poss}(\text{pickup}(x), s) \equiv \text{At}(x, s) \land \forall y \neg \text{Holding}(y, s)$
- $\text{Poss}(\text{drop}(x), s) \equiv \text{Holding}(x, s)$
- $\text{Poss}(\text{walk}(x), s) \equiv t$

Effects:

- $\text{Poss}(\text{pickup}(x), s) \supset \text{Holding}(x, \text{do}(\text{pickup}(x), s))$
- $\text{Poss}(\text{drop}(x), s) \supset \neg \text{Holding}(x, \text{do}(\text{drop}(x), s))$
- $\text{Poss}(\text{drop}(x), s) \supset \text{Broken}(x, \text{do}(\text{drop}(x), s))$
- $\text{Poss}(\text{walk}(x), s) \supset \text{At}(x, \text{do}(\text{walk}(x), s))$
- $\text{Poss}(\text{walk}(x), s) \land \text{At}(y, s) \land y \neq x \supset \neg \text{At}(y, \text{do}(\text{walk}(x), s))$

Planning vs. Programming

While regression planning works for small domains, the search space becomes unmanageable in larger, more realistic applications.

We will now consider a more realistic alternative, where we allow the user to

- prespecify (program) how to achieve a task in as much detail as desired, yet, at the same time
- leave some flexibility in choosing appropriate actions to the machine.
**Complex Actions**

Complex actions are macros which are ultimately defined in terms of primitive actions.

Consider the problem of clearing the table in a blocksworld setting. Can we write something like

\[
\text{while } (\exists \text{block}) \text{ ontable(block) do remove\_a\_block end-while}
\]

\[
\text{proc remove\_a\_block(\Pi x)[pickup(x); putaway(x)] end-proc}
\]

I.e. we do not care about the situation argument. We would like this technical point to be dealt with in the following macros (therefore the notation \(a[s]\)).

**Complex Actions in Sit.-Calc. (1)**

\(DO(a, s, s')\) stands for:

The execution of action \(a\) in situation \(s\) leads to situation \(s'\).

- **Primitive actions:**
  \[DO(a, s, s') = \text{Poss}(a[s], s) \land s' = \text{do}(a[s], s)\]

  **What does \(a[s]\) mean?** Suppose \(a\) is \(\text{read}(\text{favorite\_book(kjeld})\) and \(\text{favorite\_book}\) is a functional fluent. Then \(a[s]\) is \(\text{read}(\text{favorite\_book(kjeld, s)})\).

- **Sequence:**
  \[DO([a_1; a_2], s, s') = \exists s'' DO(a_1, s, s'') \land DO(a_2, s'', s')\]

- **Test actions:**
  \[DO(\phi, s, s') = \text{if } \phi \text{ true in the current situation?}\]

- **If-then-else, While-loops**

- **Nondeterministic actions:**
  \[DO([a_1 | a_2], s, s') = DO(a_1, s, s') \lor DO(a_2, s, s')\]

- **Nondeterministic choice:**
  \[DO((\Pi x)a(x), s, s') = \exists x DO(a(x), s, s')\]

With that we obtain a programming language for the high-level control of a robot.
7 Programming Languages for MAS
7.2 (Con-)Golog

Test actions:

\[ \text{DO}(\phi, s, s') = \phi[s] \land s = s' \]

Here \( \phi \) stands for a situation calculus formula with all situation arguments suppressed. \( \phi[s] \) stands for the arguments restored: if \( \phi \) is \( \forall x (\text{ontable}(x) \land \neg \text{on}(x, A)) \) then \( \phi[s] \) is the formula:

\( \forall x (\text{ontable}(x, s) \land \neg \text{on}(x, A, s)) \).

If \( \phi \) is \( \exists x (\text{on}(x, \text{favorite_block}(\text{nils}))) \) then \( \phi[s] \) is the formula:

\( \exists x (\text{on}(x, \text{favorite_block}(\text{nils}, s), s)) \).

Complex Actions in Sit.-Calc. (2)

If-then-else:

\[ \text{DO}([\phi, a_1, a_2], s, s') = \text{DO}([[\phi]; [a_1]], [\neg \phi]; [a_2]), s, s') \]

\* operator: execute the action \( n \) times \((n \geq 0)\):

\[ \text{DO}(a^*, s, s') = \forall P [\forall s_1 P(s_1, s_1) \land \forall s_1, s_2, s_3 (P(s_1, s_2) \land \text{DO}(a, s_2, s_3) \supset P(s_1, s_3))] \supset P(s, s') \]

While-loops:

\[ \text{DO}((\langle \phi, a \rangle, s, s') = \text{DO}([[\phi]; a]^*; \neg \phi), s, s') \]

Complex Actions in Sit.-Calc. (3)

Procedures: definition also uses \( * \) (needed because of recursion).

Here are two examples. We define \text{above}(x, y)\) as the transitive closure of \text{on}(x, y):

\[ \text{proc above}(x, y) \]

\[ (x = y) \lor [\Pi z \text{on}(x, z) \lor \text{above}(x, z)] \]

end-proc

We define \text{clean} (no arguments) as an action to put away all blocks in the box:

\[ \text{proc clean} \]

\[ (\forall x [\text{block}(x) \supset \text{in}(x, \text{box})] \lor \Pi x \forall y \neg \text{on}(y, x) \lor \text{put}(x, \text{box})] \]

end-proc

Note: \( * \)-operator needs quantification over predicates (2nd order logic). But the semantics of complex actions is completely described within the situation calculus.
Golog: A Programming Language

Golog = alGO in LOGic (Semantics = Sit.-Calc.)
It generalizes conventional languages:

■ Procedures in “normal” imperative languages are ultimately reduced to *machine instructions* like assignment statements.
■ Procedures in Golog actions are reduced to primitive actions which refer to *actions in the real world*, like picking up objects, opening doors, moving from one room to another, etc. Instead of machine states we are interested in *world states*.

An Example Golog Program

**Action preconditions:**

\[
\text{Poss}(\text{pickup}(x), s) \equiv \forall z \neg \text{Holding}(z, s)
\]
\[
\text{Poss}(\text{putontable}(x), s) \equiv \text{Holding}(x, s)
\]
\[
\text{Poss}(\text{putonfloor}(x), s) \equiv \text{Holding}(x, s)
\]

**Initial state:**

\[
\text{OnTable}(x, S_0) \equiv (x = A) \lor (x = B)
\]
\[
\forall x \neg \text{Holding}(x, S_0)
\]
### Successor State Axioms:

\[
\text{Poss}(a, s) \supset [\text{Holding}(x, \text{do}(a, s)) \equiv \neg \text{pickup}(x) \lor \text{Holding}(x, s) \land \neg \text{putonfloor}(x)]
\]

\[
\text{Poss}(a, s) \supset [\text{OnTable}(x, \text{do}(a, s)) \equiv \neg \text{putonfloor}(x) \lor \text{OnTable}(x, s) \land \neg \text{pickup}(x)]
\]

\[
\text{Poss}(a, s) \supset [\text{OnFloor}(x, \text{do}(a, s)) \equiv \neg \text{putonfloor}(x) \lor \text{OnFloor}(x, s) \land \neg \text{pickup}(x)]
\]

### Complex actions:

**proc clear_table**

\begin{align*}
\text{while } & [\exists \text{block OnTable(block)}] \text{ do} \\
& (\Pi \text{b}) \text{OnTable(b)} ?; \text{remove_from_table(b)}. \\
\text{proc remove_from_table(block)} \\
& \text{pickup(block)} ; \text{putonfloor(block)}.
\end{align*}

The evaluation of `clear_table`:

\[
\text{Axioms } \models \exists s \text{ DO(clear_table, } S_0, s) .
\]

The result is a binding of a term (action sequence) to \( s \).

### For example:

\[
s \models \text{do(putonfloor(B)}, \\
\text{do(pickup(B)),} \\
\text{do(putonfloor(A)}, \\
\text{do(pickup(A), S_0))}).
\]

[Note: The order of the blocks is not predetermined by the program.]

In general the evaluation of a Golog procedure results in a trace of the primitive actions to be executed.

---

**A simple Forward Planner in Golog**

So far we have seen how Golog allows the user to tell the system in detail how to solve a task. Golog can also be used to leave it to the system to solve a goal in the original sense of planning.
The following procedure wspdf(n) finds plans of length at most n by forward search (Reiter et al. 2000):

```plaintext
proc wspdf(n)
    goal? | [n > 0?] ; (πa)(primitive_action(a)?; a);
    ¬badSituation? ; wspdf(n - 1)
```

badSituation(s) is a domain dependent predicate used to define when a situation s should be discarded from further consideration (important for pruning the search space). For simple definitions of badSituation, this can solve problems with plans of length around 20.

We present a first-order temporal logic based on discrete, linear models with finite past and infinite future, called FML [Fisher (1992)].

FML introduces two new connectives to classical logic,

- until $(\phi \mathcal{U} \psi)$ meaning $\psi$ will become true at some future time point $t$ and in all states between and different from now and $t$, $\phi$ will be true) and
- since $(\phi \mathcal{S} \psi)$ meaning $\psi$ became true at some time point $t$ in the past and in all states between $t$ and now $\phi$ is true).
Syntax of FML.

The formulae for FML are generated as usual, starting from a set \( L_p \) of predicate symbols, a set \( L_v \) of variable symbols, a set \( L_c \) of constant symbols, the quantifiers \( \forall \) and \( \exists \), and the set \( L_t \) of terms (constants and variables). The set \( Fml \) is defined by:

- If \( t_1, ..., t_n \) are in \( L_t \) and \( p \) is a predicate symbol of arity \( n \), then \( p(t_1, ..., t_n) \) is in \( Fml \).
- \( \top \) (true) and \( \bot \) (false) are in \( Fml \).
- If \( A \) and \( B \) are in \( Fml \), then so are \( \neg A \), \( A \wedge B \), \( A \cup B \), \( A \wedge B \), and \( (A) \).
- If \( A \) is in \( Fml \) and \( v \) is in \( L_v \), then \( \exists v.A \) and \( \forall v.A \) are both in \( Fml \).

Temporal formulae can be classified as follows.

State-formula: is either a literal or a boolean combination of other state-formulae.

Strict future-time: If \( A \) and \( B \) are either state or strict future-time formulae, then \( A \cup B \) is a strict future-time formula.

Strict past-time: past-time duals of strict future-time formulae.

Non-strict: include state-formulae in their definition.

Semantics of FML.

The models for FML formulae are given by a structure consisting of a sequence of worlds (also called states), together with an assignment of truth values to atomic sentences within states, a domain \( D \) which is assumed to be constant for every state, and mappings from elements of the language into denotations.

This is like in Definition ?? on slide ??.
Definition 7.7 (FML model)

A **FML-model** is a tuple $M = \langle \sigma, D, h_c, h_p \rangle$ where

- $\sigma$ is the ordered set of states $s_0, s_1, s_2, \ldots$,
- $h_c$ is a map from the constants into $D$,
- $h_p$ is a map from $\mathbb{N} \times \mathcal{L}_p$ into $D^n \rightarrow \{\text{true}, \text{false}\}$ (the first argument of $h_p$ is the index $i$ of the state $s_i$).

Thus, for a particular state $s$, and a particular predicate $p$ of arity $n$, $h(s, p)$ assigns truth values to atoms constructed from $n$-tuples of elements of $D$.

**Definition 7.7 (FML model)**

A **FML-model** is a tuple $M = \langle \sigma, D, h_c, h_p \rangle$ where

- $\sigma$ is the ordered set of states $s_0, s_1, s_2, \ldots$,
- $h_c$ is a map from the constants into $D$,
- $h_p$ is a map from $\mathbb{N} \times \mathcal{L}_p$ into $D^n \rightarrow \{ \text{true}, \text{false} \}$ (the first argument of $h_p$ is the index $i$ of the state $s_i$).

Thus, for a particular state $s$, and a particular predicate $p$ of arity $n$, $h(s, p)$ assigns truth values to atoms constructed from $n$-tuples of elements of $D$.

**Variable assignment:** $h_v$ is a mapping from the variables into elements of $D$.

**Term assignment:** Given a variable and the valuation function $h_c$, a term assignment $\tau_{vh}$ is a mapping from terms into $D$ defined in the usual way.

**Semantics:** The semantics of FML is given by the $\models$ relation that assigns the truth value of a formula in a model $M$ at a particular moment in time $i$ and with respect to a variable assignment.

|= $T$

|= $p(x)$ iff $h_p(i, p) = \tau_{vh}(x_1), \ldots, \tau_{vh}(x_n) = \text{true}$

|= $\neg \phi$ iff $\langle M, i, h_v \rangle \models \phi$

|= $\phi \lor \psi$ iff $\langle M, i, h_v \rangle \models \phi$ or $\langle M, i, h_v \rangle \models \psi$

|= $\phi U \psi$ iff for some $k$ s.t. $i < k$, $\langle M, k, h_v \rangle \models \psi$

for all $j$, if $i < j < k$ then $\langle M, j, h_v \rangle \models \phi$

|= $\phi S \psi$ iff for some $k$ s.t. $0 \leq k < i$, $\langle M, k, h_v \rangle \models \psi$

for all $j$, if $k < j < i$ then $\langle M, j, h_v \rangle \models \phi$

|= $\forall x. \phi$ iff for all $d \in D$, $\langle M, i, h_v[d/x] \rangle \models \phi$

|= $\exists x. \phi$ iff there exists $d \in D$ s.t. $\langle M, i, h_v[d/x] \rangle \models \phi$

Concurrent METATEM is a programming language for DAI based on FML ([Fisher and Barringer (1991), Fisher (1993), Fisher and Wooldridge (1993)]).

- A Concurrent METATEM system contains a number of concurrently executing agents which are able to communicate through message passing.
- Each agent executes a first-order temporal logic specification of its desired behavior.
- Each agent has two main components:
  - an **interface** which defines how the agent may interact with its environment (i.e., other agents);
  - a **computational engine**, defining how the agent may act.
An agent interface consists of three components:

- a unique agent identifier which names the agent
- a set of predicates defining what messages will be accepted by the agent – they are called environment predicates;
- a set of predicates defining messages that the agent may send—these are called component predicates.

Besides environment and component predicates, an agent has a set of internal predicates with no external effect.

The computational engine of an object is based on the METATEM paradigm of executable temporal logics. The idea behind this approach is to directly execute a declarative agent specification given as a set of program rules which are temporal logic formulae of the form:

\[
\text{antecedent about past} \rightarrow \text{consequent about future}
\]

The intuitive interpretation of such a rule is

on the basis of the past, do the future.

Since METATEM rules must respect the past implies future form, FML formulae defining agent rules must be transformed into this form. This is always possible as demonstrated in [Barringer et al. (1989)].
Comparison via a running example

The following is taken from [Mascardi et al. (2004)].

![Diagram of buyer and seller interactions](image)

- Seller agent may receive a `contractProposal` message from a buyer agent.
- According to the **amount** of merchandise required and the **price** proposed by the buyer, the seller may **accept** the proposal, **refuse** it or try to **negotiate** a new price by sending a `contractProposal` message back to the buyer.
- The buyer agent can do the same (**accept**, **refuse** or **negotiate**) when it receives a `contractProposal` message back from the seller.

**Behaviour of seller**

if the received message is `contractProposal`(*merchandise, amount, proposedprice*)

then

- if there is enough merchandise in the warehouse and the price is greater or equal than a **max** value, the seller accepts by sending an **accept** message to the buyer and **concurrently ships** the required merchandise to the buyer (if no concurrent actions are available, answering and shipping merchandise will be executed sequentially);

- Notation used in this figure is based on an **agent-oriented extension** of UML [Parunak and Odell (2000), Odell et al. (2000)];
- the diamond with the × inside represents a “xor” connector;
- the protocol can be **repeated more than once** (note the bottom arrow from the buyer to the seller labelled with a “contractProposal” message, which loops around and up to point higher up to the seller time line).
if there is not enough merchandise in the warehouse or the price is lower or equal than a min value, the seller agent refuses by sending a refuse message to the buyer; if there is enough merchandise in the warehouse and the price is between min and max, the seller sends a contractProposal to the buyer with a proposed price evaluated as the mean of the price proposed by the buyer and max.

The merchandise to be exchanged are oranges, with minimum and maximum price 1 and 2 euro respectively. The initial amount of oranges that the seller possesses is 1000.

GOLOG

■ Primitive actions declaration:

\[ \text{ship}(\text{Buyer}, \text{Merchandise}, \text{Required-amount}) \]

The seller agent delivers the Required-amount of Merchandise to the Buyer.

\[ \text{send}(\text{Sender}, \text{Receiver}, \text{Message}) \]

Sender sends Message to Receiver.

■ Situation independent functions declaration:

\[ \text{min-price}(\text{Merchandise}) = \text{Min} \]

The minimum price the seller is willing to take under consideration for Merchandise is Min.

\[ \text{max-price}(\text{Merchandise}) = \text{Max} \]

The price for Merchandise that the seller accepts without negotiation is equal or greater than Max.

■ Primitive fluents declaration:

\[ \text{receiving}(\text{Sender}, \text{Receiver}, \text{Message}, S) \]

Receiver receives Message from Sender in situation S.

\[ \text{storing}(\text{Merchandise}, \text{Amount}, S) \]

The seller stores Amount of Merchandise in situation S.

■ Initial situation axioms:

\[ \text{min-price(orange)} = 1 \]

\[ \text{max-price(orange)} = 2 \]

\[ \forall S, R, M. \neg \text{receiving}(S, R, M, s_0) \]

\[ \text{storing}(\text{orange}, 1000, s_0) \]
Precondition axioms:

\( \text{poss}(\text{ship}(\text{Buyer, Merchandise, Required-amount}), S) \equiv \) storing(\text{Merchandise, Amount, S}) \land \text{Amount} \geq \text{Required-amount} \)

It is possible to ship merchandise iff there is enough merchandise stored in the warehouse.

\( \text{poss}(\text{send}(\text{Sender, Receiver, Message}), S) \equiv \top \)

It is always possible to send messages.

Successor state axioms:

\( \text{receiving}(\text{Sender, Receiver, Message}, \text{do}(\text{send}(\text{Sender, Receiver, Message}), S)) \equiv \top \)

Receiver receives Message from Sender in do(send(Sender, Receiver, Message), S)

reached by executing send(Sender, Receiver, Message) in S.

storing(\text{Merchandise, Amount, do(A, S)}) \equiv \)

\((A = \text{ship}(\text{Buyer, Merchandise, Required-amount}) \land \) storing(\text{Merchandise, Required-amount + Amount, S}))

\lor (A \neq \text{ship}(\text{Buyer, Merchandise, Required-amount}) \land \) storing(\text{Merchandise, Amount, S}))

The seller has a certain Amount of Merchandise if it had Required-amount + Amount of Merchandise in the previous situation and it shipped Required-amount of Merchandise, or if it had Amount of Merchandise in the previous situation and it did not ship any Merchandise.

We may think that a buyer agent executes a buyer-life-cycle procedure concurrently with the seller agent procedure seller-life-cycle.

buyer-life-cycle defines the actions the buyer agent takes according to its internal state and the messages it receives.

The seller-life-cycle is defined in the following way.
7 Programming Languages for MAS

7.4 Comparison

Concurrent MetaTeM

The Concurrent METATEM program for the seller agent may be as follows:

The interface of the seller agent is the following:

\[
\text{seller} \left( \text{contractProposal} \right) \left[ \text{accept, refuse, contractProposal, ship} \right]
\]

meaning that:

– the seller agent, identified by the seller identifier, is able to recognize a contractProposal message with its arguments, not specified in the interface;

– the messages that the seller agent is able to broadcast to the environment, including both communicative acts and actions on the environment, are accept, refuse, contractProposal, ship.

The program rules of the seller agent are the following ones (lowercase symbols = constants, uppercase = variables):

\[
\forall \text{Buyer, Merchandise, Req_Amnt, Price} \left\lbrack \text{contractProposal} \left( \text{Buyer, seller, Merchandise, Req_Amnt, Price} \right) \land \text{storing} \left( \text{Merchandise, Old_Amount} \right) \land \text{Old_Amount} \geq \text{Req_Amnt} \land \text{max - price(Merchandise, Max)} \land \text{Price} \geq \text{Max} \right\rbrack \implies \left[ \text{ship} \left( \text{Buyer, Merchandise, Req_Amnt, Price} \right) \land \text{accept} \left( \text{seller, Buyer, Merchandise, Req_Amnt, Price} \right) \right]
\]

If there was a previous state where Buyer sent a contractProposal message to seller, and in that previous state all the conditions were met to accept the proposal, then accept the Buyer’s proposal and ship the required merchandise.

The internal knowledge base of the seller agent contains the following rigid predicates (predicates whose value never changes):

- \text{min-price(orange, 1)}.
- \text{max-price(orange, 2)}.

The internal knowledge base of the seller agent contains the following flexible predicates (predicates whose value changes over time):

- \text{storing(orange, 1000)}.
∀Buyer, Merchandise, Req_Amnt, Price

\( \neg \text{contractProposal}(Buyer, seller, Merchandise, Req_Amnt, Price) \wedge \text{storing}(Merchandise, Old_Amount) \wedge \\
\text{min} = \text{price}(Merchandise, Min) \wedge \\
\text{Old_Amount} < \text{Req_Amnt} \vee \text{Price} \geq \text{Min} \implies \\
\text{refuse}(seller, Buyer, Merchandise, Req_Amnt, Price) \)

If there was a previous state where Buyer sent a contractProposal message to seller, and in that previous state the conditions were not met to accept the Buyer’s proposal, then send a refuse message to Buyer.

∀Buyer, Merchandise, Req_Amnt, Price

\( \text{contractProposal}(Buyer, seller, Merchandise, Req_Amnt, Price) \wedge \\
\text{storing}(Merchandise, Old_Amount) \wedge \text{min} = \text{price}(Merchandise, Min) \wedge \\
\text{Old_Amount} < \text{Req_Amnt} \vee \text{Price} \geq \text{Min} \implies \\
\text{contractProposal}(seller, Buyer, Merchandise, Req_Amnt, New_Price) \)

If there was a previous state where Buyer sent a contractProposal message to seller, and in that previous state the conditions were met to send a contractProposal back to Buyer, then send a contractProposal message to Buyer with a new proposed price.

7.5 References
METATEM: A framework for programming in temporal logic.

Concolog, a concurrent programming language based on the situation calculus.
*Artificial Intelligence* 121, 109–169.

METATEM at work: Modelling reactive systems using executable temporal logic.

A normal form for first-order temporal formulae.

Concurrent METATEM – A language for modeling reactive systems.
In A. Bode, M. Reeve, and G. Wolf (Eds.), *Proceedings of the 5th International Conference on Parallel Architectures and Language, Europe (PARLE’93)*, pp. 185–196. Springer-Verlag. LNCS 694.
7 Programming Languages for MAS

7.5 References

Concurrent METATEM processes – A language for distributed AI.

Executable temporal logic for distributed AI.

FIPA ACL message structure specification.
Approved for standard, 2002-12-06.
http://www.fipa.org/specs/fipa00061/.

Logic-Based Specification Environments for Multi-Agent Systems.
Ph. D. thesis, Computer Science Department of Genova University, Genova, Italy.
DISI-TH-2002-04.

Logic-based specification languages for intelligent software agents.

Extending UML for agents.
In G. Wagner, Y. Lespérance, and E. Yu (Eds.), Proceedings of the 2nd Workshop on Agent-Oriented Information Systems (AOIS’00). Held in conjunction with 17th National conference on Artificial Intelligence (AAAI’00), pp. 3–17.

Representing agent interaction protocols in UML.