Using weighted constraints to build a tutoring system for logic programming

Doctoral Thesis submitted to
Department of Informatics, University of Hamburg
by Nguyen-Thinh Le

Principal supervisor
Prof. Wolfgang Menzel
University of Hamburg

Second supervisor
Prof. Niels Pinkwart
Technical University of Clausthal

Agenda

1. Motivation: Tutoring systems for programming
2. Existing approaches
3. Approach proposal
4. Evaluation
5. Conclusions

Tutoring Systems

System

Student

Problem tasks →

← Solution attempts

Feedback & correction hints →

← Corrected solutions

The Domain of Programming

- Requirements:
  - Design a program in free form
  - Refine a program iteratively
  - The solution space is large.

Problems

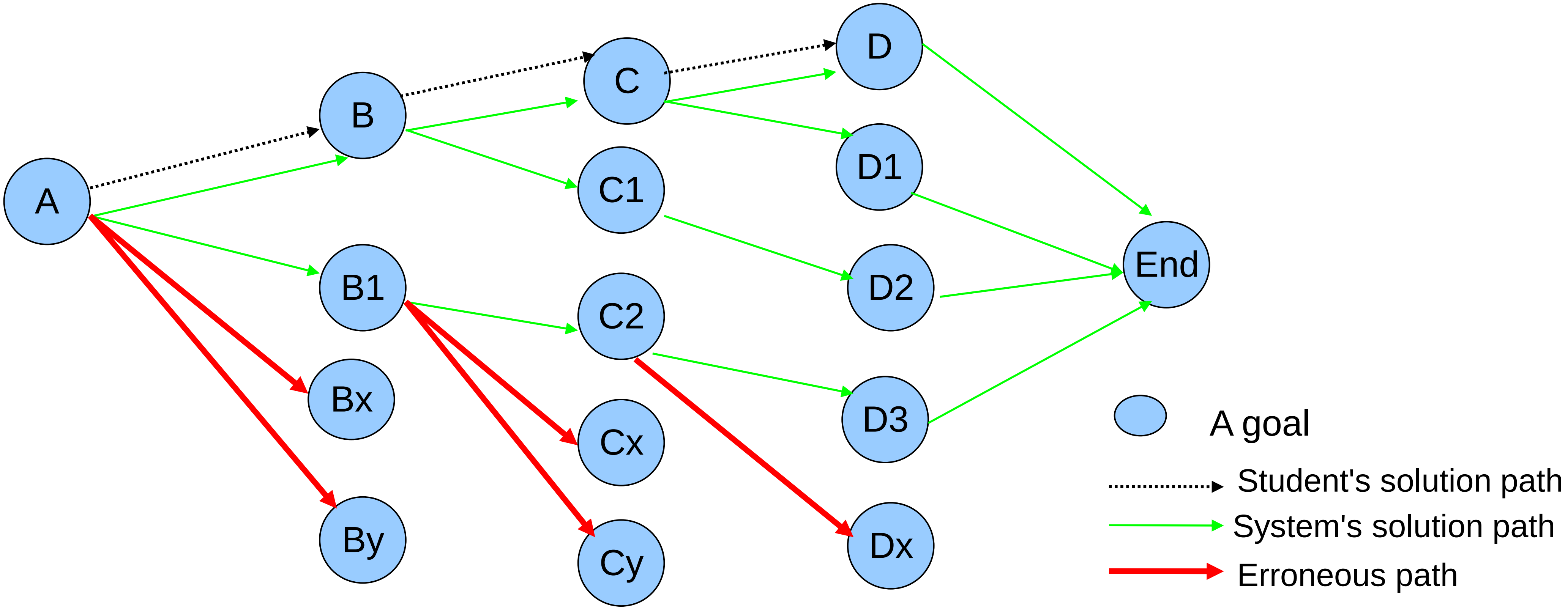
- How can a student solution be interpreted by the system correctly?
- How can the system be able to generate comprehensive feedback to the student solution?

=> The system must be modelled with sufficient domain knowledge.

Cognitive Modeling Approaches

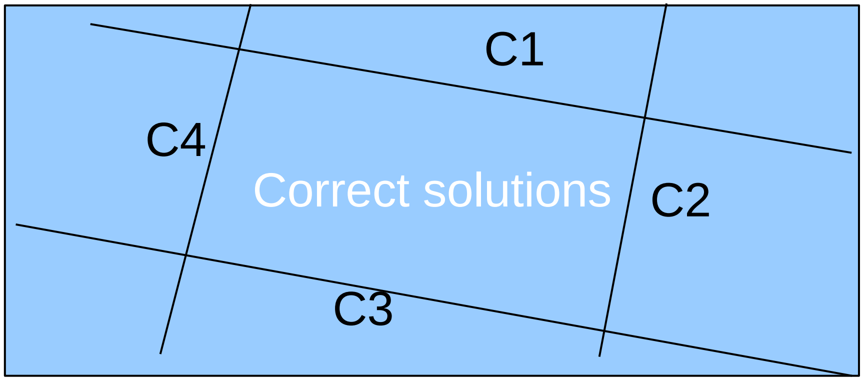
- Model tracing:
  - learning from the process of solving a problem
  - process-oriented
- Constraint-based:
  - learning from performance errors
  - product-oriented

Model-Tracing



- A solution path is represented by a sequence of problem solving steps.
- An error occurs if the student‘s solution path deviates from the system's solution path or matches an erroneous path.
- This approach tends to model linear problem solving process.

Constraint-based Modeling (CBM)



Ci: constraint

- A constraint represents a domain principle or a property of correct solutions.
- Constraints span the space of correct solutions.
- A constraint violation indicates an error in the solution.
- Advantages:
  1. No enumeration of the correct solutions
  2. No anticipation of the students' erroneous behaviours
  3. Iterative problem solving process can be modelled

Research Question

- Is the constraint-based approach applicable to build a tutoring system for the domain of logic programming?

The Solution Space

For a problem, the solution space is created on 2 levels:

1. **Solution strategy**
   - Experts have some kinds of knowledge about problem categories and solution strategies(Hoc88).
2. **Implementation**
   - Based on the available constructs of a particular domain

Alternative Solution Strategies

- Task sample: *"Write a predicate to compute the return of investment after a period for a fixed interest rate"*

Strategy	**Normal form implementation**
Analytic	inv(Sum, Rate, Period, Return) :- Return is Sum*(Rate+1)^Period.
Recursive & Arithmetic_Before	inv(Sum, _, Period, Sum) :- Period=0. inv(Sum, Rate, Period, Return) :- Period>0, NewPeriod is Period-1, inv(Sum, Rate, NewPeriod, NewSum), Return is NewSum + Rate*NewSum.
Recursive & Arithmetic_After	inv(Sum, _, Period, Sum) :- Period=0. inv(Sum, Rate, Period, Return) :- inv(Sum, Rate, NewPeriod, NewSum), Period is NewPeriod +1, Return is NewSum + Rate*NewSum.
Tail recursive	inv(Sum, _, Period, Sum):-Period=0. inv(Sum, Rate, Period, Return) :-Period>0, NewSum is Sum*Rate + Sum, NewPeriod is Period-1, inv(NewSum, Rate, NewPeriod, Return).

Alternative Implementation Variants

- Syntactic reformulations
  - Explicit or implicit unifications
  - S is M*X+N*X =>S is (M+N)*X, or S is X*N+M*X
- Alternative sequential orders
  - Clause order, subgoal order
- Defining helper predicates/functions
  - To modularize a program
- The option of introducing identifiers

Proposal 1: Semantic Table

- Goal: model alternative solution strategies
- Each solution strategy is represented by a generalised solution description (GSD).
- Arbitrary entry order => Alternative sequential orders are covered.

Str	Clause	Head	Subgoals	Description
S1	C1	p(S,R,P,Ret)	Ret is S*(R+1)^P	Use an analytic formula
S2	C1	p(S,_,P,S)	P=0	Recursion ends
S2	C2	p(S,R,P,Ret)	P>0	Check investment period
S2	C2	p(S,R,P,Ret)	NS is S*R+S	Calculate new sum
S2	C2	p(S,R,P,Ret)	NP is P-1	Update period
S2	C2	p(S,R,P,Ret)	p(NS,R,NP,Ret)	Calculate sum with new period

- S1 is a GSD for the analytic strategy; S2 is a GSD for the tail recursive strategy.

Proposal 2: Transformation rules

- Goal: cover variants of arithmetic expressions and helper predicates
- Rule 1: $A \circ X \pm B \circ X \Rightarrow (A \pm B) \circ X \mid \circ \in \{*, /\}$
- Rule 2: $A \circ B \Rightarrow B \circ A$
- Rule 3: transform user-defined helper predicates
  - Using folding/unfolding techniques
  - Limitation: not for all helper predicates

Proposal 3: Weighted Constraints

- Constraints serve the purposes:
  1. Modelling general principles of the domain, e.g.,

IF An arithmetic test subgoal comparing two variables X, Y exists

THEN Both X and Y have the instantiation state instantiated

2. Establishing a mapping between the student solution and components of the semantic table

IF in the semantic table, a component X exists and satisfies condition C

THEN in the student solution, a corresponding component must exist and satisfy condition C.

Proposal 3: Weighted Constraints

- Constraint weight represents the importance of a constraint.
- Constraint weights serve three purposes:

1. Decide for the most plausible error explanation. E.g., an arithmetic test checking whether a person is adult:

Expected solution: X >= 18 and student's input: X >=17

Possible error explanations:

- The operator is not correct. The corrective proposal is X>17
- The operand is not correct. The corrective proposal is X>=18

Constraints checking operands are more important than the ones checking operators. => The 2nd corrective proposal is more plausible.

2. Hypothesize the student's applied solution strategy
3. Ranking the severity of feedback messages

Error Diagnosis

- Hypothesis generation:
  - H(S1) = Student solution <matching> gsd (Strategy 1)
  - H(S2) = Student solution <matching> gsd (Strategy 2)
  - ...
- Hypothesis evaluation: for each H, computing the plausibility **P** by evaluating relevant constraints.
  - $P(H)=W_1*\ldots W_n$ | Wi is the weight of the i-th violated constraint
  - Max(P(H(S))) => H is the best hypothesis, S is the applied solution strategy
  - Error diagnosis is derived from constraint violations of the best hypothesis H.

Diagnostic Accuracy

- Design:
  - 221 student solutions from past written examinations
  - System's diagnosis was compared against a gold standard.

System	Pro. Language	Intention Analysis	Diagnosis Accuracy
INCOM	Prolog	87.9%	92.7%
PROUST	Pascal	81%	87%
PITS	Prolog	80%	95.8%

Learning Effects - Design

- Pre/Post test: 10 minutes
- Experiment session: 60 minutes
- Exercises are normal homework tasks not seen before.
- Questionnaire: 10 minutes
- Two studies:
  - 2009: Control (17 participants) vs. experimental group (18 participants)
  - 2010: Control (16 participants) vs. experimental group (16 participants)

Learning Effects - Results (1)

Group	Study	Learning gain (s.d.)	P-value (significant)
Control	1	0.74 (2.64)	0.27 (No)
Experimental	1	1.25 (1.81)	0.01 (Yes)
Control	2	1.28 (1.45)	0.003 (Yes)
Experimental	2	1.81 (1.79)	0.001 (Yes)

1. Groups are balanced: score(Pre, con) = score(Pre, exp)
   - p=0.07 (Study 1), p=0.48 (Study 2), sig-level=0.05
2. The learning gains: LG = score(Post) - score(Pre)
   - Experimental groups did made significant gains in both Studies
   - Only control group of Study 2 did made significant gains

Learning Effects - Results (2)

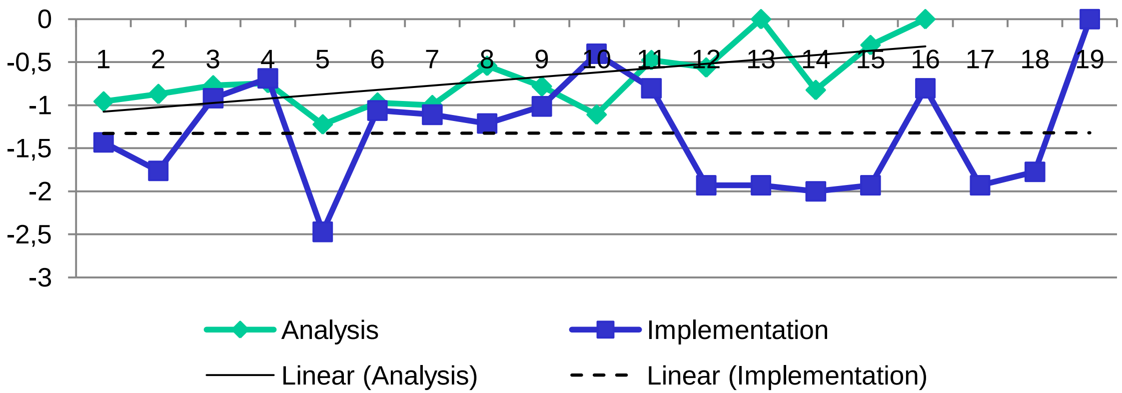
Study	Learning gain (control)	Learning gain (experimental)	p-value (significant)	Cohen's d
1	0.74 (2.64)	1.25 (1.81)	0.5 (No)	0.23
2	1.28 (1.45)	1.81 (1.79)	0.36 (No)	0.33

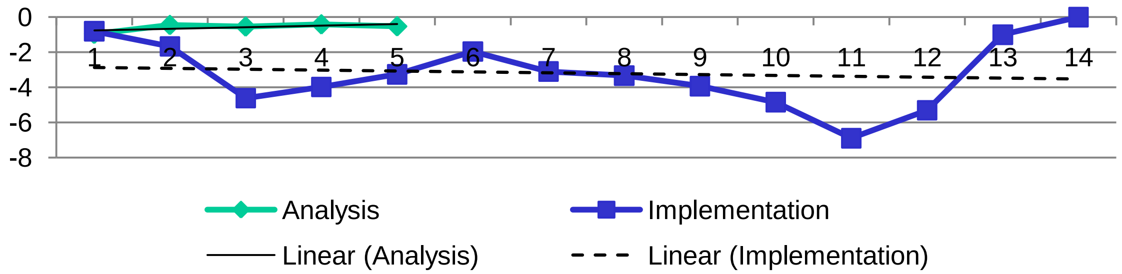
3. Is LG(exp) significantly better than LG(con)? No
4. How much is the LG(exp) better than LG(con)?

d=[0.23; 0.33] => INCOM contributed a small learning effect

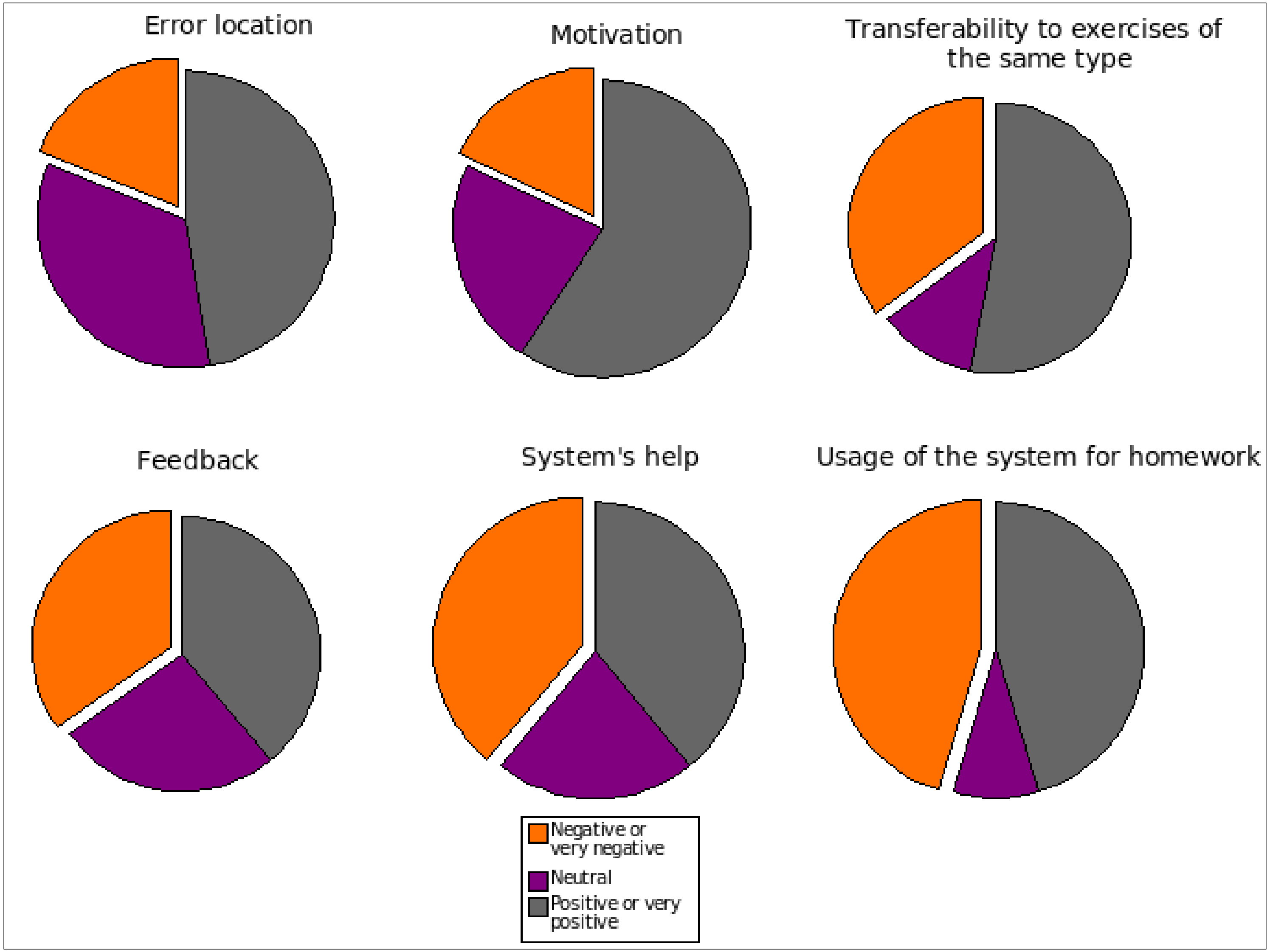
Learning Effects – Results (3)

5. Learning curves based on weighted error rates





Student's Attitudes - Results



Students confirmed
- high diagnostic accuracy
- high motivation
- confident to solve similar exercises

No clear positive answer for
- feedback comprehensiveness
- system’s helpfulness
- the usage of the system for homework

Conclusions

- Constraint weights are useful to
  - hypothesize the student intention,
  - choose the most plausible error explanation among several alternatives
  - Prioritise the severity of feedback messages
- Semantic table composes two ideas:
  - Covers alternative solution strategies
  - Represent a class of implementations in generalized form.
- Limitations:
  - A number of solution strategies need to be anticipated.
  - Not all helper predicates can be transformed.

Thank you for your attention