software competence center hagenberg

Automatic Refinement of Model Transformations

Gábor Guta

Advisors: András Pataricza, Wolfgang Schreiner, Dániel Varró

Das SCCH ist eine Initiative der

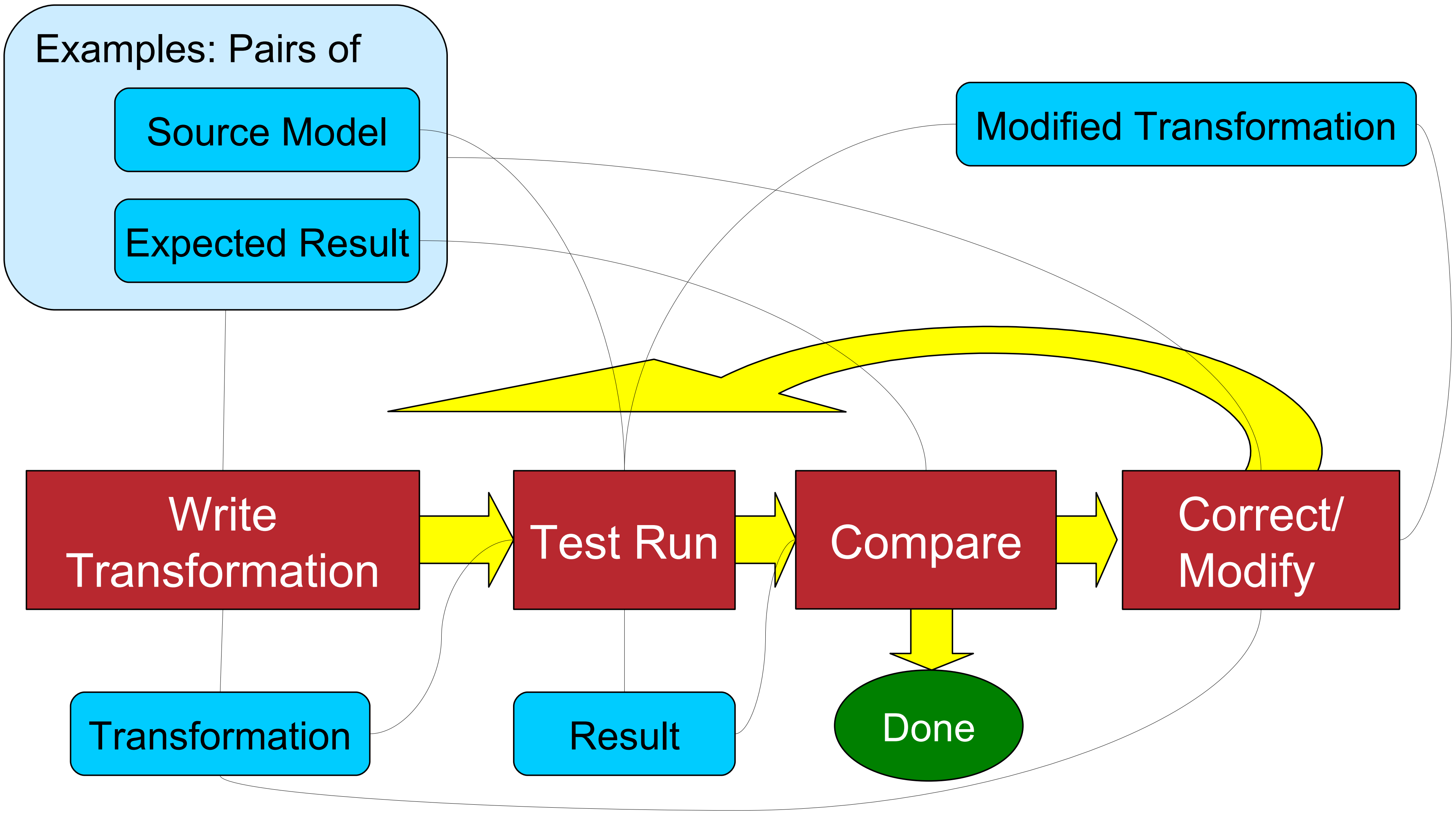
JKU JOHANNES KEPLER UNIVERSITÄT LINZ

RISC, 23.06.2010

Das SCCH befindet sich im

softwarepark hagenberg

Traditional Model Transformation Development Process



Motivation

s c c h
software competence center
hagenberg

- Aid model transformation development
  - Reduce the number and the effort of the modify/correct cycles
  - In most of the cases the modifications are trivial
- Support web page designers
- Support retargeting information to different format
  - E.g.: Source Code to Documentation

- Results may be useful in other application
  - Automatic inference of simple transformations
  - Automatic inference of domain meta-model changes
  - Quality evaluation of the transformations

The Model Transformation Modification Problem

- We have
  - a transformation (t) which generates code *G* from the source domain *S*,
  - set of examples (pairs of source models S_k and generated codes G_k),
  - modified code G_k' corresponding to each examples

```
drop procedure sp_{$TblName}_select_by_id
GO

CREATE PROCEDURE sp_${TblName}_select_by_id
   @${TblName}ID      BIGINT
AS
   SELECT
         ${TblName}ID,
         #foreach($ColumnName in $TblColumnNames)
         ${TblName}${ColumnName}#if(last!=True),#end
         #end
      FROM tbl_${TblName}
      WHERE ${TblName}ID = @${TblName}ID
GO
```

```
      FROM tbl_ItemMaster
      WHERE ItemMasterID = @ItemMasterID
GO
```

```
drop procedure sp_ItemMaster_select_by_id
GO

CREATE PROCEDURE sp_ItemMaster_select_by_id
   @ItemMasterID      BIGINT
AS
   SELECT
         ItemMasterID,
         ItemMasterNumber,
         ItemMasterDesc1,
         ItemMasterDesc2,
         ItemMasterProductTypeRef,
         ItemMasterPartTypeRef,
         ItemMasterRemark
      FROM tbl_ItemMaster
      WHERE ItemMasterID = @ItemMasterID
GO
```

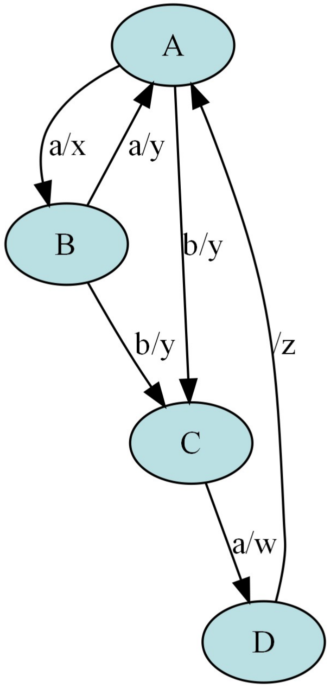
Our Approach

- Direction (Inspired by grammatical inference)
  - Developing specialized algorithms (instead of using generic optimization methods e.g. Genetic Algorithms)
  - Define measures to judge the quality of a refinement algorithms
  - Evaluate different versions of the algorithms

- Steps (Inspired by type checking solutions of XMLs)
  - Examine finite state (string) transducers
    - To investigate modification by example paradigm
  - Experiment "XSLT" like languages
    - To experiment "industrial" examples

The Example

- Transducer:



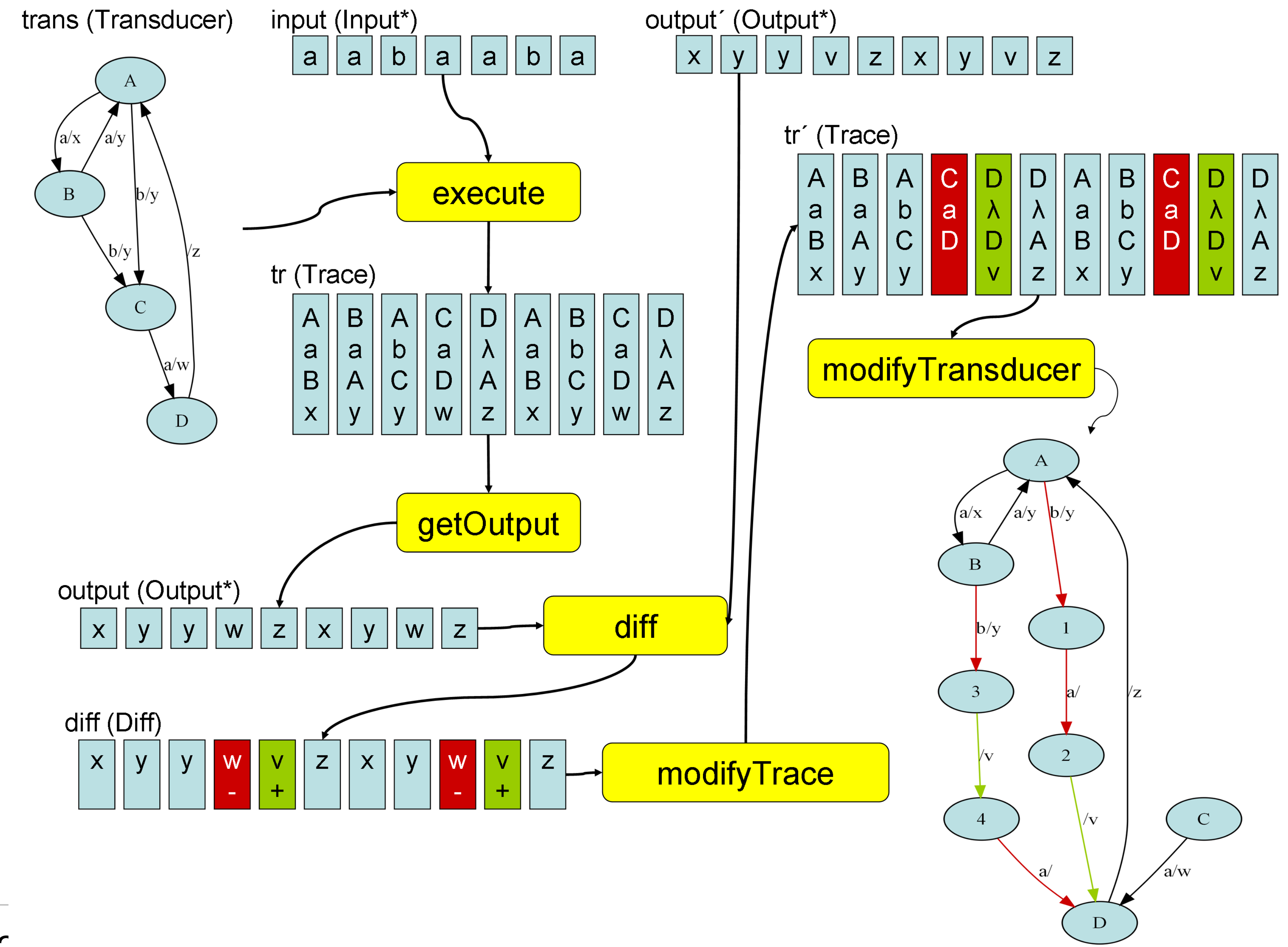
- Input: „aabaaba”
- Output: „xyywzxywz”
- Expected output: „xyyvzxyvz”

The Algorithm

- Used data types:
  - TransitionKey := (source: State, input: Input)
  - TransitionData := (target: State, output: Output)
  - Rules := TransitionKey → TransitionData
  - Transducer := (input: set(Input), output: set(Output), state: set(State), init: State, rules: Rules)
  - TraceStep := (source: State, input: Input, target: State, output: Output)
  - Trace := TraceStep*
  - Diff := (output: Output, mod: {' ', '-', '+'})*

```
inferTransducer(Transducer trans, Input* input, Output* output'):Transducer
        Trace tr=execute(trans,input)
        Output* output=getOutput(tr)
        Diff diff=calculateDiff(output,output')
        Trace tr'=modifyTrace(tr, diff, trans.init)
        Transducer trans'=modifyTransducer(trans, tr')
        return trans'
```

The Data-flow of the Example Problem



Modifying the transducer

software competence center hagenberg

```
modifyTransducer(Transducer, Trace'):Transducer
```

- If the transition in the modified trace is possible according to transition of the transducer:
  - Count the usage of the transition
- If it is not possible:
  - Add the new transition to the existing ones (only if the transition not destroy deterministic behaviour)
  - or modify an existing transition (this is the tricky part)

Modify an Existing Transition

scch software competence center hagenberg

- Modification of an existing transition (a simple version):
  - Modify the transition from the trace with a new ‘start state’
  - Modify the ‘end state’ of the preceding transition (corresponding to the preceding trace element) to the new state
  - If the transition is executed only once according to trace, we are done; Otherwise we have to modify all transitions corresponding to the preceding trace elements, until we do not find an transition executed only once

We may create several slightly different versions of the algorithm

Measuring the Efficiency of the Algorithm

s c c h
software competence center hagenberg

- Compare the result of the algorithm with a hand crafted expected result
- Structural metrics of the transducer modification
  - Number of new states
  - Number of new/modified transitions
- Behavioural metrics of the transducer modification
  - Difference in behaviour between the original transducer and modified transducer
- Metrics of the relation between the transducers and the examples
  - Coverage of the transitions

First experiments: basic metrics do not help to explain the results of the manual inspection of the results. Why?

R

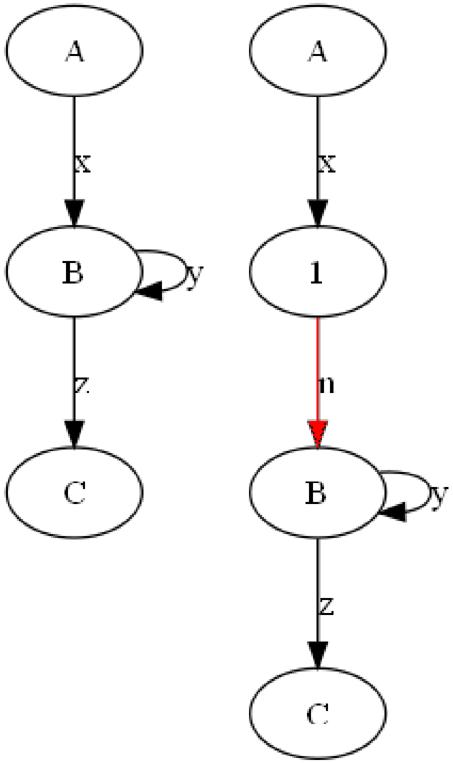
Goal-Question-Metric Paradigm

- Goal
  - Evaluate how efficiently the examples describe the modification intention of the user
- Questions
  - How many possible interpretations of the example are possible? (How clear is the intention of the user?)
  - Are the examples minimal?
  - Are the examples consistent?
- Metrics
  - Branches and cycles in the execution graph of the transducers
  - Possible interpretations of the examples

„Metrics of Intentions“

scch software competence center hagenberg

- Inserting a new edge into the graph:
- Existing examples (possible sate transitions)
  - No iteration: xnz
    A, x, 1, n, B, z / A, x, B, n, 1, z
  - One iteration: xnyz
    A, x, 1, n, B, y, B, z / A, x, B, n, 1, y, B, z
  - Two iterations: xnyyz
    A, x, 1, n, B, y, B, y, B, z



Modification Intention	Examples	Possible Interpretations	Intuition	Clarity
Before the cycle	No iteration	Before the cycle	Yes	1/2
		After the cycle	Yes	1/2
	One iteration	Before the cycle	Yes	1/2
		In the beginning of the cycle	Yes	1/2
	Two iterations	Before the cycle	Yes	1

The Concepts Behind the New Algorithm

software competence center hagenberg

- The idea is to replace the „modifyTrace“ function with a new one which
  - Does not modify the trace, but annotates the transducer
  - The annotations contains the possible way of the modification
- Then the „modifyTransducer“ will consolidate these annotations
  - The modification intention is clear
  - The modification intention is not clear
  - The annotations are contradicting

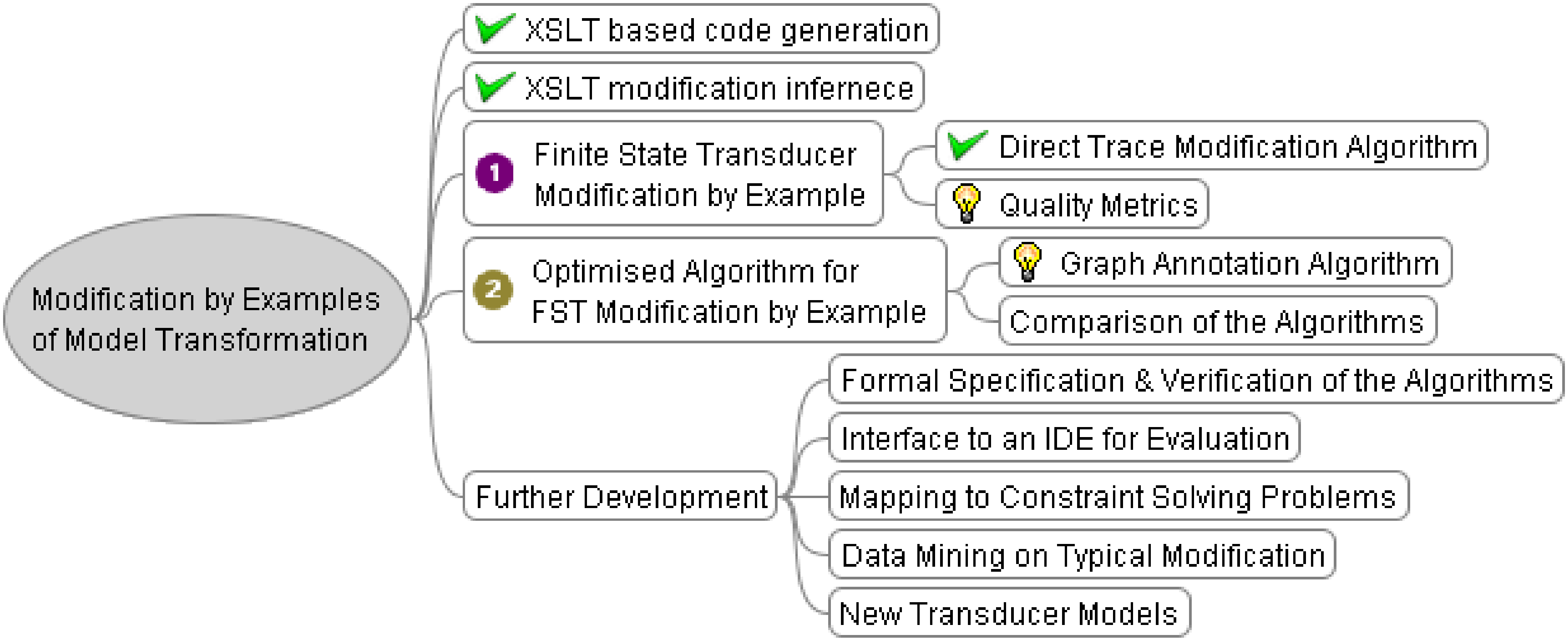
Applying the Algorithm to Transformation Languages

s c c h
software competence center
hagenberg

- Main differences in execution:
  - Transducers: selection of the transition depend on the input string
  - Transformation languages: selection of the execution path depends on the input data and the result of computed values

- Main differences in print instruction:
  - Transducers: always a fixed constant
  - Transforamtion languages: can be any computed values

Status of the Work



Conclusions

- Conclusions
  - We described an algorithm to infer transducer modifications
  - We described the concepts of a more powerful modification inference algorithm and its application concepts to transformation languages
  - We defined metrics to evaluate such algorithms
- Further work
  - Compare the algorithms