Improving Trace-Based Propagation of Feature Annotations in Model Transformations

ME Workshop 2018
Copenhagen, Danmark

Sandra Greiner and Bernhard Westfechtel
Applied Computer Science I
University of Bayreuth
Background: MVMTs

PersonDB

0..* persons

Person

name : String

Male

Female

if p.familyNotExists()
  fam <- createFamily()
  fam.insertAsParent(p)
else
  if fam.parentNotExists()
    fam.insertAsParent(p)
  else
    fam.insertAsChild(p)

Family

name : String

0..* families

Member

name : String

0..1 mum

0..* daughters

0..* sons

0..1 dad

Annotative SPLE

150% model

single-variant model transformation

Automatic propagation?

Greiner and Westfechtel – Trace-Based Propagation of Feature Annotations in Model Transformations 2
Evaluation: Commutativity

annotated multi-variant source model $m_s$

multi-variant model transformation (MVMT)

annotated multi-variant target model $m_t$

Filter

Filter

(single-variant) source model $m'_s$

single-variant model transformation

(single-variant) target model $m'_t$

(same as)

(same as)

Filter

(single-variant) target model $m''_t$
Trace-based MVMT

- No need to rewrite transformation

\( m_s \) to \( m_t \)

**SVMT**

**trace**

**transfer**

**reuse**

multi-variant source model

multi-variant target model

variability annotations

\( m_t \)

Variability annotations in model transformations.
Kind of Traces

Incomplete Trace

Generation-complete / Complete Trace

[Westfechtel and Greiner 18]
Trace-Based Propagation

Propagation based on complete traces

not always, only if rule applications are:

- **Functional**: result determined uniquely by rule match
- **Monotonic**: only adding (target) elements
- **Local**: effect only depends on match

Proved to achieve commutativity

- **Extensible**: rules applicable to same match in any model
- **Confluent**: order of rule application irrelevant

[Westfechtel and Greiner 18]
Commuting example

\[ m_s \]

\[ m_t \]

Filter

Filter

\[ DB \]

\[ pdb : PersonDB \]

\[ persons \]

\[ bs : Male \]

name = “Ben Smith”

\[ DB \land P \]

\[ as : Female \]

name = “Ann Smith”

\[ DB \land P \]

persons

Trace-based MVMT

\[ DB \]

\[ fdb : FamilyDB \]

families

\[ sf : Family \]

name = "Smith"

\[ DB \land P \]

dad

mum

\[ bs : Member \]

name = “Ben”

\[ DB \land P \]

\[ as : Member \]

name = “Ann”

\[ DB \land P \land DB \land P \]

Filter

Greiner and Westfechtel – Trace-Based Propagation of Feature Annotations in Model Transformations 7
Weaknesses – Persons to Families

Filter:

$DB \land P \land C$ to $DB \land \neg P \land C$

$pdb : PersonDB$

$bs : Male$

name = "Ben Smith"

$DB \land P$

$ts : Male$

name = "Tom Smith"

$DB \land C$

$mvmt$

$filter$

$SVMT$

$pdb : PersonDB$

$ts : Male$

name = "Tom Smith"

$DB \land P$

$fdb : FamilyDB$

$families$

$sf : Family$

name = "Smith"

$ds : Member$

name = "Ben"

$db : PersonDB$

$bs : Member$

name = "Ben"

$DB \land P$

$ts : Member$

name = "Tom"

$db : PersonDB$

$ts : Member$

name = "Tom"

$db : PersonDB$

$bs : Member$

name = "Ben"

$DB \land P$

$ abduction$ to $DB \land P \land DB \land C$
Weaknesses – Persons to Families

```r
if p.familyNotExists()
  fam <- createFamily()
  fam.insertAsParent(p)
else
  if fam.parentNotExists()
    fam.insertAsParent(p)
  else
    fam.insertAsChild(p)
```

**pdb**: PersonDB

- **bs**: Male
  - name = "Ben Smith"

- **ts**: Male
  - name = "Tom Smith"

**fdb**: FamilyDB

- **sf**: Family
  - name = "Smith"
  - sons
    - **bs**: Member
      - name = "Ben"
    - **ts**: Member
      - name = "Tom"

- **sb**: Male
  - name = "Ben Smith"

- **ts**: Male
  - name = "Tom Smith"

**pdb**: PersonDB

- **ts**: Male
  - name = "Tom Smith"

- **bs**: Male
  - name = "Ben Smith"

**fdb**: FamilyDB

- **sf**: Family
  - name = "Smith"
  - sons
    - **ts**: Member
      - name = "Tom"
    - **bs**: Member
      - name = "Ben"
1) Propagate 1:1 correspondences

- Propagation based on 1:1 correspondences
- Without considering context elements
- Annotation of Family?
2) Calculate missing annotations

Calculating missing annotations:

Element visible if

All elements directly necessary for its existence AND
At least ONE of elements needing its existence are visible
Incremental Transformations

Filter:

\[ DB \land \neg P \land C \]

\[ m_s \]

\[ pdb : PersonDB \]

\[ persons \]

\[ bs : Male \]

\[ name = "Ben Smith" \]

\[ DB \land P \]

\[ ts : Male \]

\[ name = "Tom Smith" \]

\[ DB \land C \]

\[ m'_s \]

\[ m_t \]

\[ Reused \]

\[ SVMT \]

\[ sf : Family \]

\[ name = "Smith" \]

\[ families \]

\[ bs : Member \]

\[ name = "Ben" \]

\[ dad \]

\[ ts : Member \]

\[ name = "Tom" \]

\[ sons \]

\[ Incremental \]

\[ SVMT \]

\[ m'_t \]

Greiner and Westfechtel – Trace-Based Propagation of Feature Annotations in Model Transformations 12
Commutativity

\[
\begin{align*}
\text{pdb} : \text{PersonDB} & \quad \text{DB} \\
\text{persons} & \quad \text{persons} \\
\text{bs} : \text{Male} & \quad \text{ts} : \text{Male} \\
\text{name} = \text{"Ben Smith"} & \quad \text{name} = \text{"Tom Smith"} \\
\text{DB} \land \text{P} & \quad \text{DB} \land \text{C} \\
\text{Filter:} & \quad \text{Filter:} \\
\text{DB} \land \neg \text{P} \land \text{C} & \\
\text{pdb} : \text{PersonDB} & \quad \text{fdb} : \text{FamilyDB} \\
\text{ts} : \text{Male} & \quad \text{sf} : \text{Family} \\
\text{name} = \text{"Tom Smith"} & \quad \text{name} = \text{"Smith"} \\
\text{DB} \land \text{P} & \quad \text{DB} \land \text{C} \\
\text{Incremental SVMT} & \quad \text{DB} \land \neg \text{P} \land \text{C} \\
\text{fdb} : \text{FamilyDB} & \quad \text{fdb} : \text{FamilyDB} \\
\text{families} & \quad \text{families} \\
\text{sf} : \text{Family} & \quad \text{sf} : \text{Family} \\
\text{name} = \text{"Smith"} & \quad \text{name} = \text{"Smith"} \\
\text{sons} & \quad \text{sons} \\
\text{ts} : \text{Member} & \quad \text{ts} : \text{Member} \\
\text{name} = \text{"Tom"} & \quad \text{name} = \text{"Tom"} \\
\text{bdb} : \text{PersonDB} & \quad \text{bdb} : \text{PersonDB} \\
\end{align*}
\]
Conclusion

++ With analysis of transformation specification:

(correct) propagation of annotations
++ use of most-general information
++ Solving problems with non-confluent rules

-- Open Questions – Post-process calculation:

- general strategy or heuristic? → proof?
- handling diverging execution paths in
  single- and multi-variant context