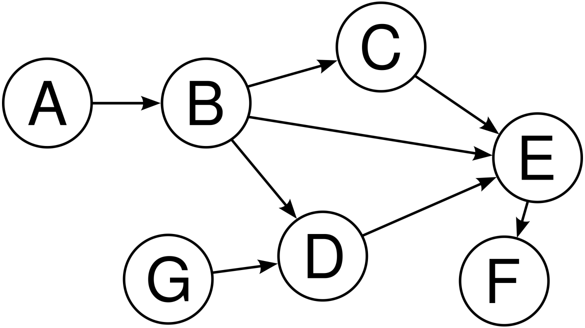


Justifications in Constraint Handling Rules for Logical Retraction in Dynamic Algorithms

Paper: Thom Frühwirth

Presentation: **Daniel Gall**

October 10, 2017

Constraint Handling Rules with Justifications

Justifications

- Mark derived information explicitly
- Track origin of information
- Logical Retraction
  - Conclusions can be withdrawn by retracting their premises

Constraint Handling Rules with Justifications

Justifications

- Mark derived information explicitly
- Track origin of information
- Logical Retraction
  - Conclusions can be withdrawn by retracting their premises

Goal

- Extend Constraint Handling Rules (CHR) with justifications ($\mathrm{CHR}^{\mathcal{J}}$)
  - Operational equivalence of rule applications
- Logical retraction
  - Correctness and confluence of retraction
- Proof-of-concept implementation

Constraint Handling Rules with Justifications – Example

Minimum

```
min(N) \ min(M) <=> N<M | true.
```

Example

$$min(1)^{\{f_1\}}, min(0)^{\{f_2\}}, min(2)^{\{f_3\}}$$

- c^F: F is set of justifications for constraint c

Constraint Handling Rules with Justifications – Example

Minimum

```
min(N) \ min(M) <=> N<M | true.
```

Example

$$min(1)^{\{f_1\}}, min(0)^{\{f_2\}}, min(2)^{\{f_3\}}$$

$$\mapsto rem(min(1)^{\{f_1\}})^{\{f_1,f_2\}}, min(2)^{\{f_3\}}, min(0)^{\{f_2\}}$$

- c^F: F is set of justifications for constraint c

Constraint Handling Rules with Justifications – Example

Minimum

```
min(N) \ min(M) <=> N<M | true.
```

Example

$$min(1)^{\{f_1\}}, min(0)^{\{f_2\}}, min(2)^{\{f_3\}}$$

$$\mapsto rem(min(1)^{\{f_1\}})^{\{f_1,f_2\}}, min(2)^{\{f_3\}}, min(0)^{\{f_2\}}$$

$$\mapsto rem(min(1)^{\{f_1\}})^{\{f_1,f_2\}}, rem(min(2)^{\{f_3\}})^{\{f_2,f_3\}}, min(0)^{\{f_2\}}$$

- c^F: F is set of justifications for constraint c

Constraint Handling Rules with Justifications – Example

Minimum

```
min(N) \ min(M) <=> N<M | true.
```

Example

$$min(1)^{\{f_1\}}, min(0)^{\{f_2\}}, min(2)^{\{f_3\}}$$

$$\mapsto rem(min(1)^{\{f_1\}})^{\{f_1,f_2\}}, min(2)^{\{f_3\}}, min(0)^{\{f_2\}}$$

$$\mapsto rem(min(1)^{\{f_1\}})^{\{f_1,f_2\}}, rem(min(2)^{\{f_3\}})^{\{f_2,f_3\}}, min(0)^{\{f_2\}}$$

- c^F: F is set of justifications for constraint c
- Constraint $min(0)$ remained
- Constraints $min(1)$ and $min(2)$ have been removed
- Constraint with justification f_2 reason for removal

Constraint Handling Rules (CHR)

- Constraints: first-order logic predicates

Constraint Handling Rules (CHR)

- Constraints: first-order logic predicates

Rules

$$H_k \setminus H_r \Leftrightarrow G \mid B.$$

- H_r removed heads (only user-defined constraints)
- H_k: kept heads (only user-defined constraints)
- G: guard (only built-in constraints)
- B: body (user-defined and built-in constraints)

- Constraints that match head and satisfy guard are removed/kept
- Body is added

CHR with Justifications (CHR$^{\mathcal{J}}$)

Original Rule

$$r : \bigwedge_{i=1}^{l} K_i \setminus \bigwedge_{j=1}^{m} R_j \Leftrightarrow C \mid \bigwedge_{k=1}^{n} B_k$$

Translated Rule

$$rf : \bigwedge_{i=1}^{l} K_i^{F_i} \setminus \bigwedge_{j=1}^{m} R_j^{F_j} \Leftrightarrow C \mid \bigwedge_{j=1}^{m} rem(R_j^{F_j})^F \wedge \bigwedge_{k=1}^{n} B_k^F$$

$$\text{where } F = \bigcup_{i=1}^{l} F_i \cup \bigcup_{j=1}^{m} F_j.$$

- F_i and F_j fresh variables that match justification sets
- Each CHR constraint in body annotated with union of all justifications

CHR with Justifications (CHR$^{\mathcal{J}}$)

Original Rule – Short Hand Notation

$$r : H_1 \setminus H_2 \Leftrightarrow C \mid B.$$

Translated Rule – Short Hand Notation

$$rf : H_1^{\mathcal{J}} \setminus H_2^{\mathcal{J}} \Leftrightarrow C \mid rem(H_2)^{\mathcal{J}} \wedge B^{\mathcal{J}}.$$

CHR with Justifications (CHR$^{\mathcal{J}}$)

Original Rule – Short Hand Notation

$$r : H_1 \setminus H_2 \Leftrightarrow C \mid B.$$

Translated Rule – Short Hand Notation

$$rf : H_1^{\mathcal{J}} \setminus H_2^{\mathcal{J}} \Leftrightarrow C \mid rem(H_2)^{\mathcal{J}} \wedge B^{\mathcal{J}}.$$

Lemma (Equivalence of Program Rules)

The following two propositions are equivalent:

- *There is a computation step with simpagation rule r: $S \mapsto_r T$.*
- *There is a computation step with justifications $S^{\mathcal{J}} \mapsto_{rf} T^{\mathcal{J}}$ with corresponding rule with justifications rf.*

Logical Retraction

Idea

- Remove CHR constraint from computation without recomputation from scratch
- All consequences due to rule applications using this constraint are undone
- Remove CHR constraints added by those rules
- Re-add CHR constraints removed by those rules

Logical Retraction

Rules for Retraction

For each constraint c/n:

$$\text{kill} : \mathit{kill}(f) \setminus G^F \Leftrightarrow f \in F \mid \mathit{true}$$

$$\text{revive} : \mathit{kill}(f) \setminus \mathit{rem}(G^{F_c})^F \Leftrightarrow f \in F \mid G^{F_c},$$

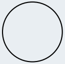
where

- $G = c(X_1, \ldots, X_n)$,
- $X_1, \ldots, X_n$ are different variables.

- Constraint may be revived and subsequently killed:
  - if F_c and F contain justification f

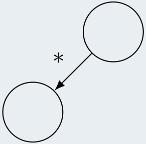
Confluence of Logical Retraction

Confluence



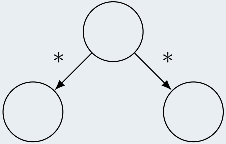
Confluence of Logical Retraction

Confluence



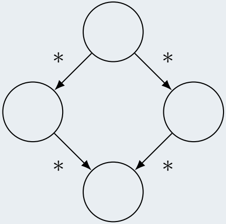
Confluence of Logical Retraction

Confluence



Confluence of Logical Retraction

Confluence



Confluence of Logical Retraction

Confluence in CHR

- Decidable criterion for terminating programs

Confluence of Logical Retraction

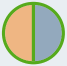
Confluence in CHR

- Decidable criterion for terminating programs
- Two rules: r_1 and r_2

Confluence of Logical Retraction

Confluence in CHR

- Decidable criterion for terminating programs
- Two rules: r_1 and r_2

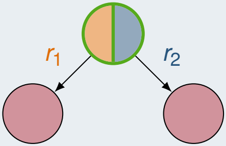


- Overlap states: Overlap heads and guard of rules

Confluence of Logical Retraction

Confluence in CHR

- Decidable criterion for terminating programs
- Two rules: r_1 and r_2

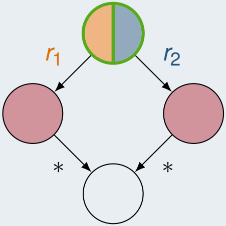


- Overlap states: Overlap heads and guard of rules
- Critical pairs: Apply rules to overlap state

Confluence of Logical Retraction

Confluence in CHR

- Decidable criterion for terminating programs
- Two rules: r_1 and r_2



- Overlap states: Overlap heads and guard of rules
- Critical pairs: Apply rules to overlap state
- If all critical pairs joinable, the program is confluent

Confluence of Logical Retraction

Idea

- Intuitively: Rules for retraction do not interefere with each other
- Additional rules do not break potential confluence of a program
- It does not make a difference if a justification is retracted immediately or if other rules are applied first

Confluence of Logical Retraction

Idea

- Intuitively: Rules for retraction do not interefere with each other
- Additional rules do not break potential confluence of a program
- It does not make a difference if a justification is retracted immediately or if other rules are applied first

Theorem (Confluence of Logical Retraction)

- *CHR program translated to rules with justifications together with kill and revive rules*
- *At most one kill(f) constraint for each justification f in any state*
- *Critical pairs between kill and revive joinable*
- *Critical pairs of those rules with any translated rule with justifications joinable*

Confluence of Logical Retraction

Proof Idea: kill and translated rules

$$\mathrm{kill} : \mathit{kill}(f) \setminus G^F \Leftrightarrow f \in F \mid \mathit{true}.$$

$$\mathit{rf} : K^{\mathcal{J}} \setminus R^{\mathcal{J}} \Leftrightarrow C \mid \mathit{rem}(R)^{\mathcal{J}} \wedge B^{\mathcal{J}}.$$

$$\mathit{kill}(f) \wedge K^{\mathcal{J}} \wedge R^{\mathcal{J}} \wedge E$$

□

Confluence of Logical Retraction

Proof Idea: kill and translated rules

$$\text{kill} : \mathit{kill}(f) \setminus G^F \Leftrightarrow f \in F \mid \mathit{true}.$$

$$\mathit{rf} : K^{\mathcal{J}} \setminus R^{\mathcal{J}} \Leftrightarrow C \mid \mathit{rem}(R)^{\mathcal{J}} \wedge B^{\mathcal{J}}.$$

$$\mathit{kill}(f) \wedge K^{\mathcal{J}} \wedge R^{\mathcal{J}} \wedge E$$

$$\xrightarrow{\mathit{kill}}$$

$$\mathit{kill}(f) \wedge E \wedge ((K^{\mathcal{J}} \wedge R^{\mathcal{J}}) - G^F)$$

$\square$

Confluence of Logical Retraction

Proof Idea: kill and translated rules

$$\text{kill} : kill(f) \setminus G^F \Leftrightarrow f \in F \mid true.$$

$$rf : K^{\mathcal{J}} \setminus R^{\mathcal{J}} \Leftrightarrow C \mid rem(R)^{\mathcal{J}} \wedge B^{\mathcal{J}}.$$

$$\begin{tikzcd}
 & kill(f) \wedge K^{\mathcal{J}} \wedge R^{\mathcal{J}} \wedge E \arrow[dl, "kill"'] \arrow[dr, "rf"] & \\
\begin{array}{c} kill(f) \wedge E \wedge \\ ((K^{\mathcal{J}} \wedge R^{\mathcal{J}}) - G^F) \end{array} & & \begin{array}{c} kill(f) \wedge E \wedge \\ K^{\mathcal{J}} \wedge rem(R)^{\mathcal{J}} \wedge B^{\mathcal{J}} \end{array}
\end{tikzcd}$$

$\square$

Confluence of Logical Retraction

Proof Idea: kill and translated rules

$$\text{kill} : \mathit{kill}(f) \setminus G^F \Leftrightarrow f \in F \mid \mathit{true}.$$

$$\mathit{rf} : K^{\mathcal{J}} \setminus R^{\mathcal{J}} \Leftrightarrow C \mid \mathit{rem}(R)^{\mathcal{J}} \wedge B^{\mathcal{J}}.$$

$$\begin{tikzcd}
 & \mathit{kill}(f) \wedge K^{\mathcal{J}} \wedge R^{\mathcal{J}} \wedge E \arrow[dl, "\mathit{kill}"'] \arrow[dr, "\mathit{rf}"] & \\
\begin{array}{c}\mathit{kill}(f) \wedge E \wedge \\ ((K^{\mathcal{J}} \wedge R^{\mathcal{J}}) - G^F)\end{array} & & \begin{array}{c}\mathit{kill}(f) \wedge E \wedge \\ K^{\mathcal{J}} \wedge \mathit{rem}(R)^{\mathcal{J}} \wedge B^{\mathcal{J}}\end{array} \arrow[ll, "\mathit{revive}^{*}, \mathit{kill}^{*}"]
\end{tikzcd}$$

$\square$

Correctness of Logical Retraction

Idea

- Result of computation after retraction the same as without adding killed constraint in the first place

Correctness of Logical Retraction

Idea

- Result of computation after retraction the same as without adding killed constraint in the first place

Theorem (Correctness of Logical Retraction)

- *Given computation where f does not occur in $A^{\mathcal{J}}$:*

  $A^{\mathcal{J}} \wedge G^{\{f\}} \wedge kill(f)$

Correctness of Logical Retraction

Idea

- Result of computation after retraction the same as without adding killed constraint in the first place

Theorem (Correctness of Logical Retraction)

- *Given computation where f does not occur in $A^{\mathcal{J}}$:*

  $A^{\mathcal{J}} \wedge G^{\{f\}} \wedge kill(f) \mapsto^* B^{\mathcal{J}} \wedge rem(R)^{\mathcal{J}} \wedge kill(f)$

Correctness of Logical Retraction

Idea

- Result of computation after retraction the same as without adding killed constraint in the first place

Theorem (Correctness of Logical Retraction)

- *Given computation where f does not occur in $A^{\mathcal{J}}$:*

$$A^{\mathcal{J}} \wedge G^{\{f\}} \wedge kill(f) \ \mapsto^* \ B^{\mathcal{J}} \wedge rem(R)^{\mathcal{J}} \wedge kill(f) \ \not\mapsto_{kill,revive}$$

Correctness of Logical Retraction

Idea

- Result of computation after retraction the same as without adding killed constraint in the first place

Theorem (Correctness of Logical Retraction)

- *Given computation where f does not occur in $A^{\mathcal{J}}$:*

  $A^{\mathcal{J}} \wedge G^{\{f\}} \wedge kill(f) \; \mapsto^* \; B^{\mathcal{J}} \wedge rem(R)^{\mathcal{J}} \wedge kill(f) \; \not\mapsto_{kill,revive}$

- *Then there is computation without $G^{\{f\}}$:*

  $A^{\mathcal{J}} \; \mapsto^* \; B^{\mathcal{J}} \wedge rem(R)^{\mathcal{J}}$

Correctness of Logical Retraction

Proof Idea.

- Mapping between computations with a constraint $G^{\{f\}}$ and without
    - Strip away constraints that contain justification *f* except for *rem*
- For all rules:
    - show that stripped transition of rule application is equivalent to rule application without $G^{\{f\}}$

$\square$

Implementation

Basic Idea

- Apply translation scheme
- Represent justifications as unbound variables
- `C ## [F1,F2,...]`: constraint `C` with justifications `F1`, `F2`, ...
- Built-in constraint `union` computes union of justification sets
- For *kill* and *revive*: guard $f \in F$ via `member(F,Fs)`

Optimization

Thom Frühwirth: *Implementation of Logical Retraction in Constraint Handling Rules with Justifications*

Proceedings of the 21st International Conference on Applications of Declarative Programming and Knowledge Management (INAP) September 2017

Minimum Example

Translated Minimum Rule

```
min(A)##B \ min(C)##D <=> A<C |
    union([B,D],E), rem(min(C)##D)##E.
```

Example

```
?- min(1)##[A], min(0)##[B], min(2)##[C].
rem(min(1)##[A])##[A,B], rem(min(2)##[C])##[B,C],
min(0)##[B].
```

- Constraint `min(0)` remained
- Constraints `min(1)` and `min(2)` have been removed
- `0` is minimum
- Constraint with justification `B` reason for removal

Minimum Example

Translated Minimum Rule

```
min(A)##B \ min(C)##D <=> A<C |
    union([B,D],E), rem(min(C)##D)##E.
```

Example

```
?- min(1)##[A], min(0)##[B], min(2)##[C],
      killc(min(0)).
rem(min(2)##[C])##[A,C], min(1)##[A].
```

- Logically retract current minimum `min(0)`
- Constraint `min(0)` is removed by binding justification `B`.
- The `rem` constraints for `min(1)` and `min(2)` involve `B` as well
- The two constraints are re-introduced and react with each other
- `min(2)` is now removed by `min(1)`

Conclusion

- Source-to-source transformation for CHR with justifications ($\mathrm{CHR}^{\mathcal{J}}$)
- Logical retraction of constraints
- Correctness theorem: If constraint is retracted, computation continues as if constraint was never there
- Confluence theorem: Implementation of retraction with two-rule scheme is confluent
- Online translator:

  `http://pmx.informatik.uni-ulm.de/chr/translator/`

Future Work

- Investigate behavior of logical and classical algorithms with justifications
- Programs not required to be confluent, but use with non-confluent programs may lead to unwanted orders of rule applications
- Improve implementation further, optimizations, benchmarks
- Extend rule scheme to support debugging by explanation
- For error diagnosis: detection and repair of inconsistencies

Thank you for your attention!

Questions?