

Abstract Interpretation for Constraint Handling Rules

Tom Schrijvers, Peter J. Stuckey, Gregory J. Duck

July 13, 2005





- 1 Introduction to CHR
- 2 Program Analysis Problem
- 3 Abstract Interpretation Framework for CHR
- 4 Framework Instantiations
 - Groundness Analysis
 - Late Storage Analysis
- 5 Conclusion
 - Current & Future Work



- 1 Introduction to CHR
- 2 Program Analysis Problem
- 3 Abstract Interpretation Framework for CHR
- 4 Framework Instantiations
 - Groundness Analysis
 - Late Storage Analysis
- 5 Conclusion
 - Current & Future Work

Introduction to Constraint Handling Rules (CHR)

4/22

What is CHR (Thom Frühwirth)?

- ▶ input: constraints (e.g. $X \leq Y$)
- ▶ program: conditional rewriting rules
- ▶ output: solved form, after exhaustive rule application
- ▶ how: don't worry (unspecified in high-level semantics)

What are the advantages of CHR?

- ▶ **what**, not **how**
- ▶ compact programs
- ▶ easy to extend and modify
- ▶ practical: **how** when necessary (refined semantics)

An Example of CHR

5/22

A Less-Than-Or-Equal-To Constraint

```
refl @ X ≤ X          <=> true.  
antisym @ X ≤ Y , Y ≤ X <=> X = Y.  
redund @ X ≤ Y \ X ≤ Y <=> true.  
trans @ X ≤ Y , Y ≤ Z ==> X ≤ Z.
```

An Example of CHR

5/22

A Less-Than-Or-Equal-To Constraint

```
refl @ X ≤ X           <=> true.  
antisym @ X ≤ Y , Y ≤ X <=> X = Y.  
redund @ X ≤ Y \ X ≤ Y <=> true.  
trans @ X ≤ Y , Y ≤ Z ==> X ≤ Z.
```

A Query

```
?- X ≤ Y, Y ≤ Z.
```

An Example of CHR

5/22

A Less-Than-Or-Equal-To Constraint

```
refl @ X ≤ X          <=> true.  
antisym @ X ≤ Y , Y ≤ X <=> X = Y.  
redund @ X ≤ Y \ X ≤ Y <=> true.  
trans @ X ≤ Y , Y ≤ Z ==> X ≤ Z.
```

A Query

```
?- X ≤ Y, Y ≤ Z.
```

The Constraint Store

An Example of CHR

5/22

A Less-Than-Or-Equal-To Constraint

```
refl @ X ≤ X          <=> true.  
antisym @ X ≤ Y , Y ≤ X <=> X = Y.  
redund @ X ≤ Y \ X ≤ Y <=> true.  
trans @ X ≤ Y , Y ≤ Z ==> X ≤ Z.
```

A Query

```
?-          Y ≤ Z.
```

The Constraint Store

```
X ≤ Y
```

An Example of CHR

5/22

A Less-Than-Or-Equal-To Constraint

```
refl @ X ≤ X          <=> true.  
antisym @ X ≤ Y , Y ≤ X <=> X = Y.  
redund @ X ≤ Y \ X ≤ Y <=> true.  
trans @ X ≤ Y , Y ≤ Z ==> X ≤ Z.
```

A Query

```
?- .
```

The Constraint Store

```
X ≤ Y, Y ≤ Z
```

An Example of CHR

5/22

A Less-Than-Or-Equal-To Constraint

```
refl @ X ≤ X          <=> true.  
antisym @ X ≤ Y , Y ≤ X <=> X = Y.  
redund @ X ≤ Y \ X ≤ Y <=> true.  
trans @ X ≤ Y , Y ≤ Z ==> X ≤ Z.
```

A Query

```
?- .
```

The Constraint Store

```
X ≤ Y, Y ≤ Z
```

An Example of CHR

5/22

A Less-Than-Or-Equal-To Constraint

```
refl @ X ≤ X <=> true.  
antisym @ X ≤ Y , Y ≤ X <=> X = Y.  
redund @ X ≤ Y \ X ≤ Y <=> true.  
trans @ X ≤ Y , Y ≤ Z ==> X ≤ Z.
```

A Query

```
?- .
```

The Constraint Store

```
X ≤ Y, Y ≤ Z, X ≤ Z
```

An Example of CHR

5/22

A Less-Than-Or-Equal-To Constraint

```
refl @ X ≤ X          <=> true.  
antisym @ X ≤ Y , Y ≤ X <=> X = Y.  
redund @ X ≤ Y \ X ≤ Y <=> true.  
trans @ X ≤ Y , Y ≤ Z ==> X ≤ Z.
```

A Query

```
?- .
```

The Constraint Store

```
X ≤ Y, Y ≤ Z, X ≤ Z
```

What is CHR for?

- ▶ custom constraint solvers
find solution to constraints
- ▶ typical application domains:
 - ▶ planning
 - ▶ scheduling
- ▶ new domains:
 - ▶ type inference and checking
 - ▶ natural language processing
 - ▶ multi-agent systems
- ▶ any general purpose application



- 1 Introduction to CHR
- 2 Program Analysis Problem**
- 3 Abstract Interpretation Framework for CHR
- 4 Framework Instantiations
 - Groundness Analysis
 - Late Storage Analysis
- 5 Conclusion
 - Current & Future Work

Performance Issues

- ▶ Performance, scalability crucial for real-world applications
- ▶ CHR is a high-level language
- ▶ high-level languages have poor performance
- ▶ yes, programmers do not control efficiency
- ▶ ...but **optimized compilation** to the rescue!

Optimized Compilation of CHR

- ▶ reasoning about high-level languages is easy
- ▶ Basis: efficient but general compilation schema by C. Holzbaur
- ▶ First generation of optimizations:
 - ▶ local to a rule
 - ▶ specialization of schema
 - ▶ based on pattern matching
- ▶ Second generation of optimizations:
 - ▶ between rules
 - ▶ ad hoc analysis of program
- ▶ Third generation: ?

Program Analysis Problem

10/22

Program Analysis Problem

- ▶ several optimizing CHR systems (Hal CHR, K.U.Leuven CHR)
- ▶ different versions of ad hoc analysis
- ▶ not documented
- ▶ not formalized

Desiderata

- ▶ exchange analyses
- ▶ compare different versions
- ▶ see opportunities for improvement
- ▶ combine analyses

Solution : Abstract Interpretation (Cousot & Cousot)

- ▶ established methodology
- ▶ language-independent
- ▶ formal description
- ▶ easy to compare, improve and combine analyses
- ▶ reach a higher level in analysis complexity

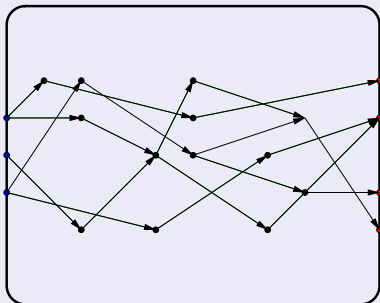


- 1 Introduction to CHR
- 2 Program Analysis Problem
- 3 Abstract Interpretation Framework for CHR**
- 4 Framework Instantiations
 - Groundness Analysis
 - Late Storage Analysis
- 5 Conclusion
 - Current & Future Work

Abstract Interpretation Approach

13/22

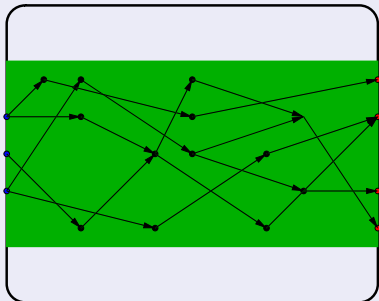
Concrete Derivations



concrete semantics:

- ▶ derivation steps
- ▶ from **initial state**
- ▶ to **final state**
- ▶ non-determinism!

Abstract Derivations



abstract semantics:

- ▶ approximation
- ▶ abstract steps
- ▶ 1 abstract state = many concrete
- ▶ capture non-determinism

Framework

- ▶ recursive formulation of semantics
- ▶ requirements for abstract states (Galois connection)
- ▶ requirements for abstract semantics

$$\begin{array}{ccc} \sigma_1 & \xrightarrow{S[[\mathcal{P}]]} & \sigma_2 \\ \alpha \downarrow & & \uparrow \gamma \\ s_1 & \xrightarrow{AS[[\mathcal{P}]]} & s_2 \end{array}$$

- ▶ specification of initial states
- ▶ default approximations for non-determinism



- 1 Introduction to CHR
- 2 Program Analysis Problem
- 3 Abstract Interpretation Framework for CHR
- 4 Framework Instantiations**
 - Groundness Analysis
 - Late Storage Analysis
- 5 Conclusion
 - Current & Future Work

Groundness Analysis

16/22

Groundness Analysis for Prolog

- ▶ classic analysis, many domains
- ▶ instantiation grounds variables
- ▶ what variables are ground at what points

```
main :-  
    ( X = Y,  
      Y = a  
    ;  
      X = b  
    ).
```

Groundness Analysis for CHR

Groundness Analysis

16/22

Groundness Analysis for Prolog

Groundness Analysis for CHR

- ▶ many benefits: hash-tables, no reactivation
- ▶ CHR does not ground variables
- ▶ underlying solver (Prolog) does!
- ▶ generic groundness analysis for CHR
- ▶ parameter = Prolog analysis (current = naive)

Late Storage Analysis

17/22

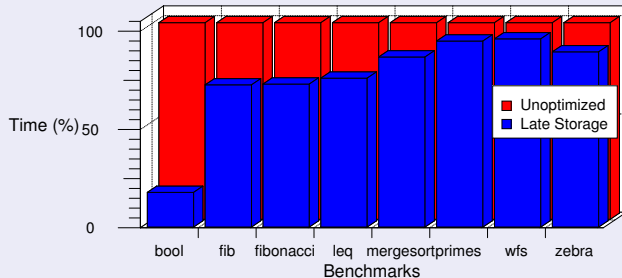
Late Storage

- ▶ delay storage of constraint
- ▶ to avoid store management overhead
- ▶ never-stored property
- ▶ simple formulation with A.I.
- ▶ see paper

Analysis Results

18/22

Late Storage



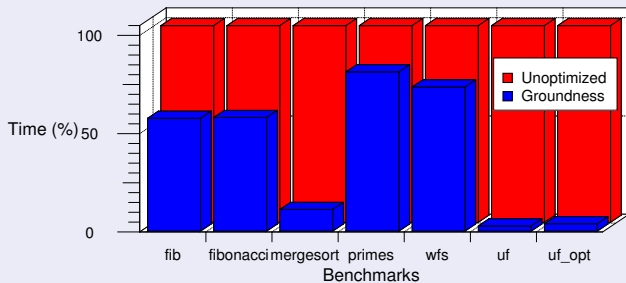
Groundness

Analysis Results

18/22

Late Storage

Groundness





- 1 Introduction to CHR
- 2 Program Analysis Problem
- 3 Abstract Interpretation Framework for CHR
- 4 Framework Instantiations
 - Groundness Analysis
 - Late Storage Analysis
- 5 Conclusion
 - Current & Future Work

Contributions:

- ▶ 1st use of abstract interpretation for CHR
- ▶ general framework for analysis
- ▶ two actual domains
 - ▶ late storage
 - ▶ groundness
- ▶ simple analysis, strong results

Current & Future Work

21/22

Current Work

- ▶ T. Schrijvers, *Analyses, Optimizations and Extensions of Constraint Handling Rules*, Ph.D. Thesis, June 10, 2005.
- ▶ G.J. Duck and T. Schrijvers, *Accurate Functional Dependency Analysis for Constraint Handling Rules*, submitted to CHR 2005.

Future Work

- ▶ reformulate more analyses
- ▶ combine and improve analyses
- ▶ study trade-offs in precision and efficiency
- ▶ couple with host language analysis

The End

22/22

Questions?

Thank you!