A linear logic semantics for CHR

Hariolf Betz
University of Ulm
Overview

1. CHR revisited
   • How is the classical declarative semantic flawed?

2. Introduction to linear logic
   • What makes linear logic a better choice for a declarative semantics?

3. A linear logic semantics for CHR
CHR overview

• „constraint handling rules“
• developed in the 1990s
• originally intended for the implementation of constraint solvers
• nowadays increasingly used as a general-purpose concurrent programming language
CHR example

A leq B <=> A=B | true.
A leq B, B leq A <=> A=B.
A leq B, B leq C ==> A leq C.

<A leq B ∧ B leq C ∧ C leq A, true>
<A leq B ∧ B leq C ∧ C leq A ∧ A leq C, true>
<A leq B ∧ B leq C, A=C>
<A=B, A=C>
<T, A=B ∧ A=C>
Operational semantics

\[ F \Leftrightarrow D \mid H \]

\[ \text{CT} \vdash \forall (C \rightarrow \exists x (F = E \land D)) \]

\[ < E \land G, C > \leftrightarrow < H \land G, (F = E) \land D \land C > \]

\[ F \iff D \mid H. \quad < E \land G, C > \]

\[ < H \land G, (F = E) \land D \land C > \]
Classical declarative Semantics

\[ E \iff C \quad \mid \quad G \]

\[ \forall (C \rightarrow (E \leftrightarrow \exists_y G)) \]

\[ E \implies C \quad \mid \quad G \]

\[ \forall (C \rightarrow (E \rightarrow \exists_y G)) \]
Problems arising: Coin example

\textbf{Throw(coin) } \iff \text{ coin=sideA.}

\forall (\text{throw(coin)} \iff \text{coin = sideA})

\textbf{Throw(coin) } \iff \text{ coin=sideB.}

\forall (\text{throw(coin)} \iff \text{coin = sideB})

\Rightarrow \forall (\text{coin=sideA} \iff \text{coin = sideB})

\Rightarrow \forall (\text{sideA = sideB})

- a direct contradiction to what we intended
Linear logic

• Introduced by Jean-Yves Girard in 1987
• Resource-oriented logic
  – one can be prohibited from re-using the same hypotheses several times
• „Logic of actions“
linear implication

- „lollipop“
- „consuming ... yields ...

- For one euro, we can buy a pizza.
  
  € $\rightarrow$ P
linear implication

- „lollipop“
- „consuming ... yields ..."

\[ A \rightarrow B \quad A \land B \]
\[ (A \rightarrow B) \land A \vdash (A \rightarrow B) \land A \land B \]
\[ A \circ B \quad A \quad B \]
\[ (A \circ B) \otimes A \vdash B \]
„times“ conjunction

- „times“
- „both ... and ...“

- For one euro, we can buy a pizza and a coke
  \[ \€ \rightarrow \mathcal{P} \otimes \mathcal{C} \]

- A pizza is two euros
  \[ \€ \otimes \€ \rightarrow \mathcal{P} \]
& „with“ conjunction

• „with“
• „choose from ... and ...“

• For one euro, we can buy a coke or a milkshake

€ → C & M

A&B ⊨ A
A&B ⊨ B

A&B ⊭ A ⊗ B
! „bang“ modality

- „bang“
- „of course ...“
- used to represent
  - stable facts
  - unlimited resources
- For one euro, we can buy a cup of coffee with unlimited refills

\[ \mathcal{E} \rightarrow \Diamond !C \]

\[ \vdash !C \rightarrow (\Diamond !(\neg C \otimes C)) \]
Why use linear logic for CHR?

• Similarity between CHR operational semantics and linear logic calculus
  – Consumption of resources
  – Rewriting of constraints

• Avoidance of contradictory semantics
  – Ref. coin example

• Representation of the multiset nature of CHR constraints
Linear logic declarative Semantics

\[ E \iff C \mid G \]

\[ !\forall([C] \multimap (E \multimap \exists y G)) \]

\[ E \implies C \mid G \]

\[ !\forall([C] \multimap (E \multimap E \otimes \exists y G)) \]
Back to coin throwing problem

- \texttt{throw(coin) <-> coin=sideA.}
  
  \( \forall (\text{throw(coin)} \rightarrow \text{coin=sideA}) \)

- \texttt{throw(coin) <-> coin=sideB.}
  
  \( \forall (\text{throw(coin)} \rightarrow \text{coin=sideB}) \)

  \( \forall (\text{throw(coin)} \rightarrow \text{coin=sideA ~sideB}) \)
Conclusion

• The classical declarative semantics of CHR aims at constraint solvers only
• When used as a general-purpose concurrent programming language some flaws arise
  – Contradictory semantics
  – Flawed representation of redundant resources
• A linear logic declarative semantics is closer to the operational semantics