# On Automating the Calculus of Relations

Peter Höfner and Georg Struth

Universität Augsburg

August 12, 2008

# Relations

- one of the most ubiquitous concepts in mathematics and computing
- origins in the late 19th century
- 1941: The calculus of (binary) relations (A.Tarski)
- first-order, equational axioms

# Relation Calculi

**Applications**

- program semantics (Dijkstra, Hoare,...)
- refinement calculus (Back, Scott,...)
- verification

**Relation-based Formal Methods**

- Alloy (Jackson)
- B (Abrial)
- Z (Spivey)
- algebraic approach to functional programming (Bird, de Moor)

**Further Applications**

- data bases, graphs, preference modelling, modal reasoning, linguistics, hardware verification, design of algorithms, ...

# Relations

- a binary relation $R$ on a set $A$ is a subset of $A \times A$ (a set of ordered pairs)
- operations
  — union $R \cup S$
  — intersection $R \cap S$
  — complement $\overline{R}$
  — relative product $R;S$
  $(a,b) \in R;S \Leftrightarrow \exists c.\ (a,c) \in R$ and $(c,b) \in S$
  — converse $\breve{R}$ $\quad (a,b) \in \breve{R} \Leftrightarrow (b,a) \in R$
- $(2^{A^2}, \cup, ;, \bar{\ }, \breve{\ }, 1_A)$ is called proper relation algebra of all binary relations
- expressiveness of the calculus of binary relations is that of the three-variable fragment of first-order logic

# Relation algebra

### Definition

A relation algebra is a structure $(A, +, ;, ^{-}, ^{\smile}, 1)$ satisfying the axioms

$$(x+y)+z = x+(y+z)\ , \qquad x+y = y+x\ , \qquad x = \overline{\overline{x}+\overline{y}} + \overline{\overline{x}+y}\ ,$$
$$(x;y);z = x;(y;z)\ , \qquad (x+y);z = x;z+y;z\ , \qquad x;1 = x\ ,$$
$$\breve{\breve{x}} = x\ , \qquad (x+y)^{\smile} = \breve{x}+\breve{y}\ , \qquad \breve{x};\overline{x;y}+\overline{y} = \overline{y}\ .$$

- meet can be defined as $x \cdot y = \overline{\overline{x}+\overline{y}}$
- a partial order is given by $x \leq y \Leftrightarrow x+y = y$

- a relation algebra is representable iff it is isomorphic to a proper one
- too weak to prove some truths about binary relations
- but: translation into logic can introduce quite complex expressions with nested quantifiers and destroy the inherent algebraic structure
- equational theory is undecidable

# On Automating the Calculus of Relations

- interactive proof-checkers (von Oheimb, Kahl)
- special-purpose proof systems, e.g.,
  — tableaux calculi (Maddux)
  — Rasiowa-Sikorski calculus (Orlowska)
- translation into the (undecidable) fragment of predicate logic (SPASS 3.0)

**Why not use off-the-shelf theorem provers combined with Tarski's equational axioms?**

# Results and Experience

- more than $100$ theorems proved as base library
- most of them without difficulties
- some needed restriction of axioms or additional hypothesis
  Axiom selection systems seem necessary (e.g., SRASS)
- Prover9/Waldmeister perform best
  (evaluation of more than $10$ ATP systems)
- a comparison between our approach and translation into predicate logic is still missing

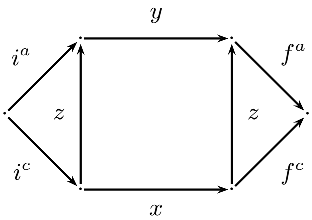
# Simulation Laws for Data Refinement

- program refinement investigates the stepwise transformation of abstract specifications to executable code
- data refinement is a variant that considers the transformation of *abstract* data types (ADTs) into *concrete* ADTs

**Abstract ADTs**

- observed through the effects of their operations on states
- operations are usually modelled as binary relations
- further operations model the initialisation and finalisation of ADTs

# Simulations




## Definition (de Roever, Engelhardt)

Let $x$, $y$ and $z$ be elements of some relation algebra.

- $x$ *U-simulates* $y$ with respect to $z$ ($x \subseteq^z_U y$) if $\breve{z};x;z \leq y$,
- $x$ *L-simulates* $y$ with respect to $z$ ($x \subseteq^z_L y$) if $\breve{z};x \leq y;\breve{z}$,
- $x$ $\breve{U}$*-simulates* $y$ with respect to $z$ ($x \subseteq^z_{\breve{U}} y$) if $x \leq z;y;\breve{z}$,
- $x$ $\breve{L}$*-simulates* $y$ with respect to $z$ ($x \subseteq^z_{\breve{L}} y$) if $x;z \leq z;y$.

($z$ is the abstraction relation; $\subseteq$ the simulation relation)

# Data Refinement

**Theorem (soundness of simulations)**

- $L$- *and* $\breve{L}$-*simulations are sound for data refinement*
- $U$-*simulations are sound if the simulation relation is total* $(1 \leq x;\breve{x})$
- $\breve{U}$-*simulations are sound if the simulation relation is a function* $(\breve{x};x \leq 1)$

**Remarks**

- the proof uses structural induction
- the entire induction cannot be treated by ATP systems
- but: all base cases and induction steps can be proven fully automatically

# Stepwise Proof for $L$-simulation

**base cases**

- $0 \subseteq_L^z 0$ and $1 \subseteq_L^z 1$
  (Prover9: $< 10\,s$)
- the case of atomic operations holds by assumption

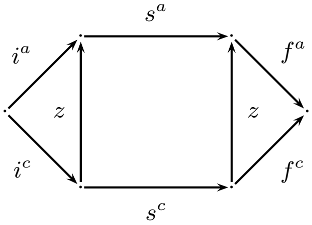
**induction step**

Let $s_1^c \subseteq_L^z s_1^a$ and $s_2^c \subseteq_L^z s_2^a$.

- composition: $s_1^c ; s_2^c \subseteq_L^z s_1^a ; s_2^a$
  (Prover9: $< 3\,s$)
- choice: $s_1^c + s_2^c \subseteq_L^z s_1^a + s_2^a$
  (Prover9: $< 2\,s$ using an additional distributivity law)
- iteration: $(s_1^c)^* \subseteq_L^z (s_1^a)^*$
  (Prover9: $< 1\,s$)

  ($^*$ is the reflexive, transitive closure and can be axiomatised in first-order logic)

# Stepwise Proof for $L$-simulation




**final step** Let $i^c \leq i^a ; \breve{z}$, $\breve{z} ; f^c \leq f^a$ and $s^c \subseteq_L^z s^a$

- $i^c ; s^c ; f^c \leq i^a ; s^a ; f^a$
  (Prover9: $< 1\,s$)

# Conclusion

- combination of relation algebra and ATP systems is feasible
- ATP systems can speed up finding proofs / counterexamples
  (We found flaws in the soundness proof for $U$ and $\breve{U}$-simulations)
- alternative higher-order, special-purpose, translational and finitist approaches
- examples suggest that formal methods become more automatic
- practical verification tasks often require the integration of algebraic techniques into a wider context:
  Most induction proofs require higher-order reasoning, but the base case and the induction step can often be discharged algebraically.

# Outlook

- results hopefully pave the way for interesting applications in relational software development methods like B, Z or Alloy
- relations are not only used for ADTs, e.g.,
  weakest liberal precondition (wlp$(x, p) = \overline{x;\overline{p}}$) or
  weakest precondition
- find ways of combining the abstract pointfree level with the concrete data
- integration of ordered chaining techniques (Bachmair, Ganzinger) into modern ATP systems would make relational reasoning more efficiently
- a combination with hypothesis learning techniques seems indispensable for tackling more complex applications and larger specifications