

Bounded Relational Analysis of Free Data Types

A. Dunets S. Schellhorn W. Reif

Department of Computer Science
University of Augsburg

The Second International Conference on Tests and Proofs
Prato, Italy
April 10, 2008

Motivation

- Unsuccessful proof attempts of **wrong conjectures**

Motivation

- Unsuccessful proof attempts of **wrong conjectures**
- **Why wrong?**

Motivation

- Unsuccessful proof attempts of **wrong conjectures**
- **Why wrong?**
- **Finite model finding** can help
 - support for proof engineer
 - saving time and effort

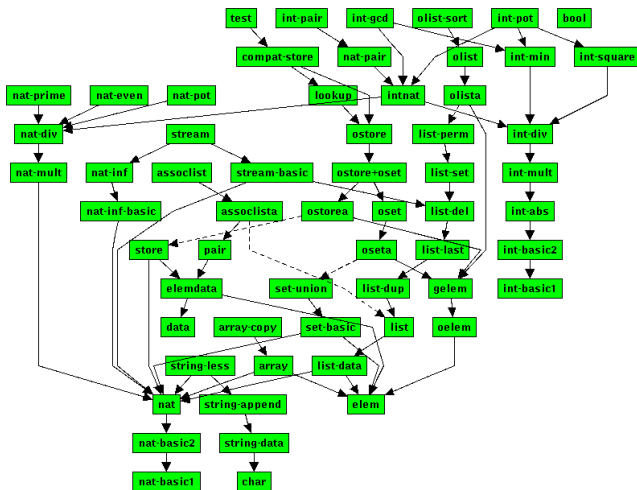
Motivation

- Unsuccessful proof attempts of **wrong conjectures**
- **Why wrong?**
- **Finite model finding** can help
 - support for proof engineer
 - saving time and effort
- **Challenge:**
 - Which class of specifications?
 - What kind of finite models?
 - Which class of sentences?
 - How?

Outline

- 1 Introduction**
 - Theorem Prover KIV
 - Alloy Analyzer
- 2 Finite Models**
 - Construction
 - Compatible Definitions
- 3 Bounded Analysis**
 - Amenable Sentences
 - Example: Lists of Intervals
 - More Results
- 4 Summary**
 - Conclusions & Outlook

Algebraic Specifications - KIV library.



Algebraic Specification of Lists.

Data Specification

```
generic data specification
  parameter elem

  list = [] | . + . ( . .first : elem; . .rest : list);

  variables x, y, z : list;
end generic data specification
```

Enrichment

```
enrich list with
  functions
    . + . : list x list -> list;
    rev   : list -> list;
  axioms
    app-nil  : [] + x = x;
    app-cons : (a + x) + y = a + (x + y);
    rev-nil  : rev([]) = [];
    rev-cons : rev(a + x) = rev(x) + (a + []);
end enrich
```

Algebraic Specifications.

Key notions:

- *specification*
 - $SP = (\Sigma, Ax, Gen)$

Algebraic Specifications.

Key notions:

- *specification*
 - $SP = (\Sigma, Ax, Gen)$
- *model*
 - $\mathcal{A} \models SP \Leftrightarrow \mathcal{A} \in Alg(\Sigma), \mathcal{A} \models Gen, \mathcal{A} \models Ax$

Algebraic Specifications.

Key notions:

- *specification*
 - $SP = (\Sigma, Ax, Gen)$
- *model*
 - $\mathcal{A} \models SP \Leftrightarrow \mathcal{A} \in Alg(\Sigma), \mathcal{A} \models Gen, \mathcal{A} \models Ax$
- *freely generated term algebras*
 - syntactically different terms denote different values

Algebraic Specification of Lists.

specification $SP = (\Sigma, Ax, Gen)$

- $\Sigma = (\{\text{elem}, \text{list}\}, \{[], +, \text{.first}, \text{.rest}\} \cup \{+, \text{rev}\})$
- $Ax = \{\text{app-nil}, \text{app-cons}, \text{rev-nil}, \text{rev-cons}\}$
- $Gen = \{\text{list} = [] \mid . + . (\text{.first} : \text{elem}; \text{.rest} : \text{list})\}$

Algebraic Specification of Lists.

specification $SP = (\Sigma, Ax, Gen)$

- $\Sigma = (\{\text{elem}, \text{list}\}, \{[], +, \text{.first}, \text{.rest}\} \cup \{+, \text{rev}\})$
- $Ax = \{\text{app-nil}, \text{app-cons}, \text{rev-nil}, \text{rev-cons}\}$
- $Gen = \{\text{list} = [] \mid . + . (\text{.first} : \text{elem}; \text{.rest} : \text{list})\}$

model

- $\mathcal{A} = ((\mathcal{A}_s)_{s \in \{\text{elem}, \text{list}\}}, (f_{\mathcal{A}})_{f \in \{[], +, \dots\}})$
- $\mathcal{A}_{\text{elem}} = \{a, b, c, \dots\}$
- $\mathcal{A}_{\text{list}} = \{[], [a], [b], \dots, [a, a], [a, b], \dots\}$
- $\text{rev}_{\mathcal{A}}([a, b]) = [b, a]$

Algebraic Specification of Lists.

specification $SP = (\Sigma, Ax, Gen)$

- $\Sigma = (\{\text{elem}, \text{list}\}, \{[], +, \text{.first}, \text{.rest}\} \cup \{+, \text{rev}\})$
- $Ax = \{\text{app-nil}, \text{app-cons}, \text{rev-nil}, \text{rev-cons}\}$
- $Gen = \{\text{list} = [] \mid . + . (\text{.first} : \text{elem}; \text{.rest} : \text{list})\}$

model

- $\mathcal{A} = ((\mathcal{A}_s)_{s \in \{\text{elem}, \text{list}\}}, (f_{\mathcal{A}})_{f \in \{[], +, \dots\}})$
- $\mathcal{A}_{\text{elem}} = \{a, b, c, \dots\}$
- $\mathcal{A}_{\text{list}} = \{[], [a], [b], \dots, [a, a], [a, b], \dots\}$
- $\text{rev}_{\mathcal{A}}([a, b]) = [b, a]$

Challenge:

- generate finite cutouts of \mathcal{A} with Alloy

Automating Relational Logic

[Jackson, 2000] @ MIT

Alloy's specification SP

- Finite structure
 - **finite multisorted universe**
 - finite set of atoms for each sort: D_1, \dots, D_n
 - predicates (relations) on atoms
- Sentence φ (**with quantifiers**)

Modes of operation

- **Counter example** for $SP \wedge \neg\varphi$
- **Witness** of $SP \wedge \varphi$

Alloy

Multisorted Relational Logic

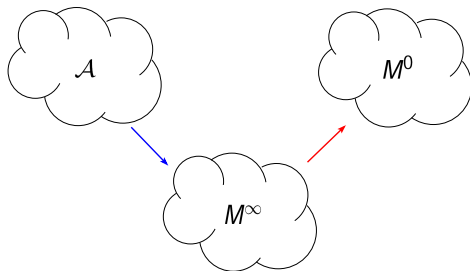
First-order relational logic with transitive closure

- Atomic formulas $A \equiv x_1 = x_2 \mid r(x_1, \dots, x_n)$
- Relational terms r
 - predicates p (relations)
 - composition $r_1.r_2$
 - transitive closure ^+r (for binary r)

Semantics

- defined on $M = (D_{S_1}, \dots, D_{S_n}, \gamma)$:
 - domains D_{S_i}
 - γ interprets predicates p on \underline{D}

Generating Finite Models in Alloy



- **challenges:**

- How to specify M^∞ and M^0 in Alloy?
- Which class of M^0 is adequate?
- Which class of φ can be checked on M^0 ?

From \mathcal{A} to M^∞

SUGA axioms generate M^∞

- for $\Sigma = (S, \bigcup_s C_s)$
[Kuncak and Jackson, 2005]

extend $SP = (\Sigma, Ax, Gen)$

- $\Sigma' = (S, \bigcup_s C_s \cup \{f_1, \dots, f_n\})$
- $Ax' = Ax \cup \{\Phi_1, \dots, \Phi_n\}$

From \mathcal{A} to M^∞

Extending Σ

- function $f : s_1 \times \dots \times s_n \rightarrow s$ to relation $F : s_1 \times \dots \times s_n \times s$
- two additional axioms:
 - **functionality**: $\forall \underline{x}, y_1, y_2. F(\underline{x}, y_1) \wedge F(\underline{x}, y_2) \rightarrow y_1 = y_2$
 - **totality**: $\forall \underline{x}. \exists y. F(\underline{x}, y)$
- relational definition Φ :

$$\forall \underline{x}, y. F(\underline{x}, y) \leftrightarrow Q_1 v_1 \dots Q_m v_m. \chi(\underline{v}, \underline{x}, y)$$

From \mathcal{A} to M^∞

Example: reverse function

- reverse function, $rev : list \times list$
 - axioms: $rev([]) = []$, $rev(a + x) = rev(x) + (a + [])$

From \mathcal{A} to M^∞

Example: reverse function

- reverse function, $rev : list \times list$
 - axioms: $rev([]) = []$, $rev(a + x) = rev(x) + (a + [])$
- transformation to relational form:

$$\forall x, y. REV(x, y) \leftrightarrow \exists a, z_1, z_2, z_3, z_4.$$

$$NIL(x) \wedge NIL(y) \vee CONS(a, z_1, x) \wedge NIL(z_3) \wedge$$

$$CONS(a, z_3, z_2) \wedge REV(z_1, z_4) \wedge APP(z_4, z_2, y)$$

From M^∞ to M^0

Closedness Constraints

- Finite cutouts from M^∞ . How?
- SUA axioms generate *subterm-closed* submodels M^0
[Kuncak and Jackson, 2005]
 - for some (f, Φ) *too weak*

From M^∞ to M^0

Closedness Constraints

- Finite cutouts from M^∞ . How?
- SUA axioms generate *subterm-closed* submodels M^0 [Kuncak and Jackson, 2005]
 - for some (f, Φ) *too weak*
- *in general*: preorder $\preceq: \bigcup D_i \times D_j$ with

$$\forall d_0 \in D^0, d \in D^\infty. d \preceq d_0 \rightarrow d \in D^0$$

From M^∞ to M^0

Closedness Constraints

- Finite cutouts from M^∞ . How?
- SUA axioms generate *subterm-closed* submodels M^0 [Kuncak and Jackson, 2005]
 - for some (f, Φ) *too weak*
- *in general*: preorder $\preceq: \bigcup D_i \times D_j$ with

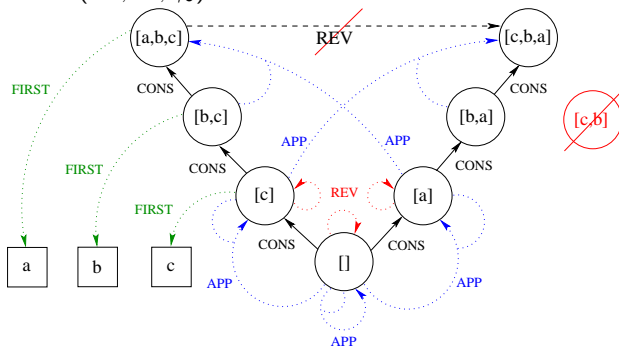
$$\forall d_0 \in D^0, d \in D^\infty. d \preceq d_0 \rightarrow d \in D^0$$

- identify *adequate* \preceq , construct \preceq -closed submodels
 - \preceq **depends on** Φ

Definitions on M^0

Subterm-closedness too weak

- $M^0 = (E^0, L^0, \gamma_0)$



- definition for *reverse*:

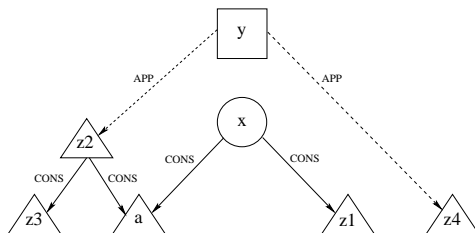
$$\forall x, y. \text{REV}(x, y) \leftrightarrow$$

$$\exists a, z_1, z_2, z_3, z_4. \text{NIL}(x) \wedge \text{NIL}(y) \vee \text{CONS}(a, z_1, x) \wedge$$

$$\text{NIL}(z_3) \wedge \text{CONS}(a, z_3, z_2) \wedge \text{REV}(z_1, z_4) \wedge \text{APP}(z_4, z_2, y)$$

Definitions on M^0

Closedness Constraints



- definition for *reverse*:

$$\forall x, y. REV(x, y) \leftrightarrow \exists a, z_1, z_2, z_3, z_4.$$

$$NIL(x) \wedge NIL(y) \vee CONS(a, z_1, x) \wedge NIL(z_3) \wedge \\ CONS(a, z_3, z_2) \wedge REV(z_1, z_4) \wedge APP(z_4, z_2, y)$$

- additional constraint:

$$\forall y. \exists a, z_1, z_2, z_3. NIL(y) \vee APP(z_1, z_2, y) \wedge \\ CONS(a, z_3, z_2) \wedge NIL(z_3)$$

Definitions on M^0

Closedness Constraints

- does not work for not surjective functions (e.g. *palindrome*)

Definitions on M^0

Closedness Constraints

- does not work for not surjective functions (e.g. *palindrome*)
 - **stronger** constraints: *s, k-completeness*, $C_s^k(M^0)$:

$$\forall c \in C_s. \forall \underline{d} \in \underline{D}^0. (\bigwedge_{\text{sort}(d_i)=s} \#_s(d_i) < k) \rightarrow c(\underline{d}) \in D_s^0$$

Definitions on M^0

Closedness Constraints

- does not work for not surjective functions (e.g. *palindrome*)
 - **stronger** constraints: *s, k-completeness*, $\mathcal{C}_s^k(M^0)$:

$$\forall c \in \mathcal{C}_s. \forall \underline{d} \in \underline{D}^0. (\bigwedge_{\text{sort}(d_i)=s} \#_s(d_i) < k) \rightarrow c(\underline{d}) \in D_s^0$$
 - exponential growth:
 - $M^0 = (E^0, L^0, \gamma_0)$ with $E^0 = \{a, b\}$ and $\mathcal{C}_{list}^2(M^0)$
 - $L_0 = \{[], [a], [b], [a, a], [a, b], [b, a], [b, b]\}$

Definitions on M^0

Closedness Constraints

- does not work for not surjective functions (e.g. *palindrome*)
 - **stronger** constraints: *s, k-completeness*, $\mathcal{C}_s^k(M^0)$:

$$\forall c \in \mathcal{C}_s. \forall \underline{d} \in \underline{D}^0. (\bigwedge_{\text{sort}(d_i)=s} \#_s(d_i) < k) \rightarrow c(\underline{d}) \in D_s^0$$
 - exponential growth:
 - $M^0 = (E^0, L^0, \gamma_0)$ with $E^0 = \{a, b\}$ and $\mathcal{C}_{list}^2(M^0)$
 - $L_0 = \{[], [a], [b], [a, a], [a, b], [b, a], [b, b]\}$
- *artificial example: rev* using *butlast*

$$\text{rev}(a + x) = \text{butlast}(\text{rev}(x) + a + b)$$

\Rightarrow no \preceq

Definitions on M^0

Closedness Constraints

- does not work for not surjective functions (e.g. *palindrome*)
 - **stronger** constraints: *s, k-completeness*, $\mathcal{C}_s^k(M^0)$:

$$\forall c \in \mathcal{C}_s. \forall \underline{d} \in \underline{D}^0. (\bigwedge_{\text{sort}(d_i)=s} \#_s(d_i) < k) \rightarrow c(\underline{d}) \in D_s^0$$
 - exponential growth:
 - $M^0 = (E^0, L^0, \gamma_0)$ with $E^0 = \{a, b\}$ and $\mathcal{C}_{list}^2(M^0)$
 - $L_0 = \{[], [a], [b], [a, a], [a, b], [b, a], [b, b]\}$
- *artificial example: rev* using **butlast**

$$\text{rev}(a + x) = \text{butlast}(\text{rev}(x) + a + b)$$

\Rightarrow no \preceq
- **challenge**: identifying \preceq -compatible definitions

Existential and universal formulas

Finite satisfiability & refutation

- $\forall \underline{v}. \chi(\underline{v})$ (instantiation of \underline{v} refuting χ in M^0 can be used in M^∞ as well)
- $\exists \underline{v}. \chi(\underline{v})$ (witness for χ in M^0 is also a witness in M^∞)
- **not amenable** formulas - no finite counter examples/witnesses:
 - $\forall n. \exists m. m > n$
 - $\exists m. \forall n. m \leq n$

Specification

- list of free blocks of dynamically allocated memory

Specification

- list of **free blocks of dynamically allocated memory**
- set of nats **uniquely** represented by intervals:
 - set $\{0, 1, 2, 4, 5, 7\}$
 - list of intervals: $[(0, 2), (4, 5), (7, 7)]$

Specification

- list of **free blocks of dynamically allocated memory**
- set of nats **uniquely** represented by intervals:
 - set $\{0, 1, 2, 4, 5, 7\}$
 - list of intervals: $[(0, 2), (4, 5), (7, 7)]$
- **predicate R** defines well-formed lists
 - $R([(0, 2), (4, 5), (7, 7)]) = \text{true}$
 - $R([(2, 0)]) = \text{false}$, $R([(0, 2), (2, 4)]) = \text{false}$

Specification

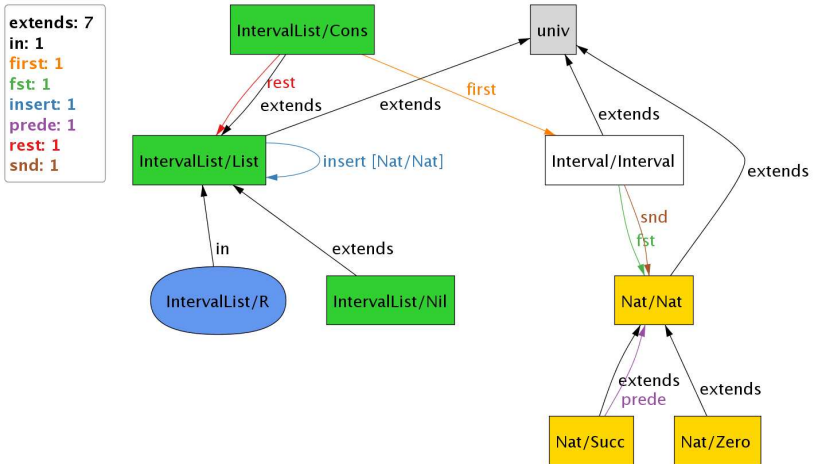
- list of **free blocks of dynamically allocated memory**
- set of nats **uniquely** represented by intervals:
 - set $\{0, 1, 2, 4, 5, 7\}$
 - list of intervals: $[(0, 2), (4, 5), (7, 7)]$
- **predicate R** defines well-formed lists
 - $R([(0, 2), (4, 5), (7, 7)]) = \text{true}$
 - $R([(2, 0)]) = \text{false}$, $R([(0, 2), (2, 4)]) = \text{false}$
- given specification of **operation insert** : $\text{ivlist} \times \text{nat} \rightarrow \text{ivlist}$
 - ? check the invariant: $\forall \text{ivl}_1, \text{ivl}_2 : \text{ivlist}, n : \text{nat}.$
 $R(\text{ivl}_1) \wedge \text{ivl}_2 = \text{insert}(\text{ivl}_1, n) \rightarrow R(\text{ivl}_2)$

Specification

- $sorts = \{nat, interval, ivlist\}$
- predicate $R : ivlist$ and function $insert : list \times nat \rightarrow list$
- *subterm-closedness* suffices for both definitions ✓
- invariant: universal sentence ✓

Example: Lists of Intervals

Metamodel in Alloy



Example: Lists of Intervals

Counter Example

- invariant violated for $insert(ivl_1, n, ivl_2)$
 - $ivl_1 = [(0, 0), (2, 2)]$
 - $ivl_2 = [(0, 1), (2, 2)]$
 - $n = 1$
- minimal scope: $|L^0| \geq 4$

More Results

KIV library

sorts	bounds	s, k-compl	#ops	#clauses	w-#vars	w-time
nat	$ N \leq 8$	-	17	3×10^5	2×10^5	9 s
	$ N \leq 10$	-	17	5×10^5	7×10^5	30 s
	$ N \leq 12$	-	17	1×10^6	1×10^6	2 min
list, elem, nat	$ L \leq 3, E = 2, N = 2$	(list,1)	43	3×10^4	1×10^4	0.5 s
	$ L \leq 4, E = 3, N = 2$	(list,1)	43	1×10^5	4×10^4	1 s
	$ L \leq 7, E = 2, N = 3$	(list,2)	43	1×10^6	4×10^5	9 s
	$ L \leq 13, E = 3, N = 3$	(list,2)	20	9×10^6	3×10^6	4 min
	$ L \leq 15, E = 2, N = 4$	(list,3)	10	12×10^6	4×10^6	> 5 min

Table: Benchmark: wrong conjectures, average case (zChaff SAT solver, 2.4 GHz Dual Core).

- size of \mathcal{M}^0 limited to $|L^0| \leq 7, |E^0| = 2, |N^0| = 3$ suffices

Conclusions

- generation of finite instances:
 - **why** the conjecture is wrong?
 - **what** is an instance of the specification?
 - source of information for designers and proof engineers

Conclusions

- generation of finite instances:
 - **why** the conjecture is wrong?
 - **what** is an instance of the specification?
 - source of information for designers and proof engineers
- open issues

Outlook

- heuristics for identifying compatible definitions
- extension to non-freely generated data types (**stores, sets, arrays**)
- bigger case studies, scalability

Thank You

Questions ?