A Proof Assistant for Alloy Specifications

Mattias Ulbrich, Ulrich Geilmann, Aboubakr Achraf El Ghazi, Mana Taghdiri

Karlsruhe Institute of Technology, Germany

TACAS 2012
Tallinn, March 30

Motivation

Alloy is a widely-used declarative modeling language

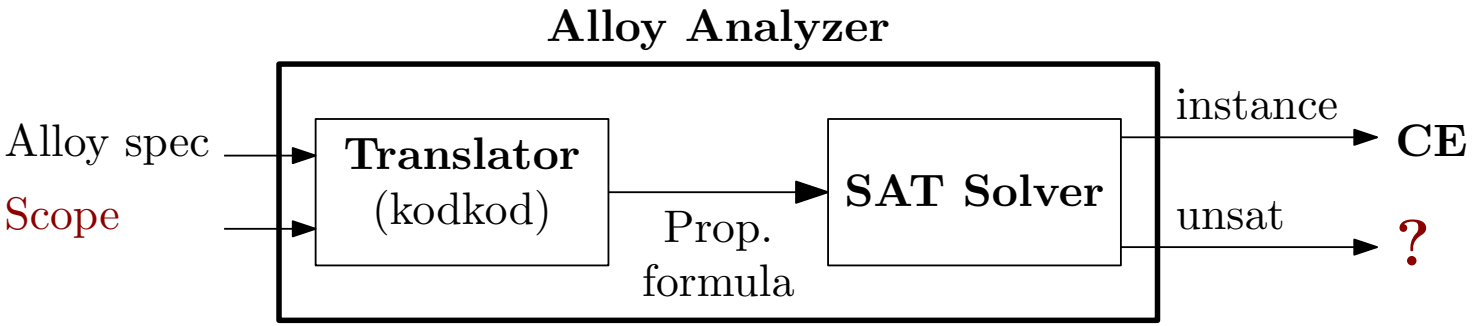
- Stand-alone framework for checking high-level system designs
- Intermediate language for checking programs
- Engine for generating counterexamples for theorem provers

The Alloy language

- Simple, uniform semantics
  - Everything is a relation
  - Semantics equivalent to first-order + relational calculus
- Expressive, familiar syntax
  - Set and relational operators, first-order quantifiers, transitive closure, linear integer arithmetic
  - Concise formulation of rich properties
  - Syntax similar to an OO language
- Analyzable
  - The Alloy Analyzer (AA) is fully automatic
  - Requires boundedness

The Alloy Analyzer (AA)

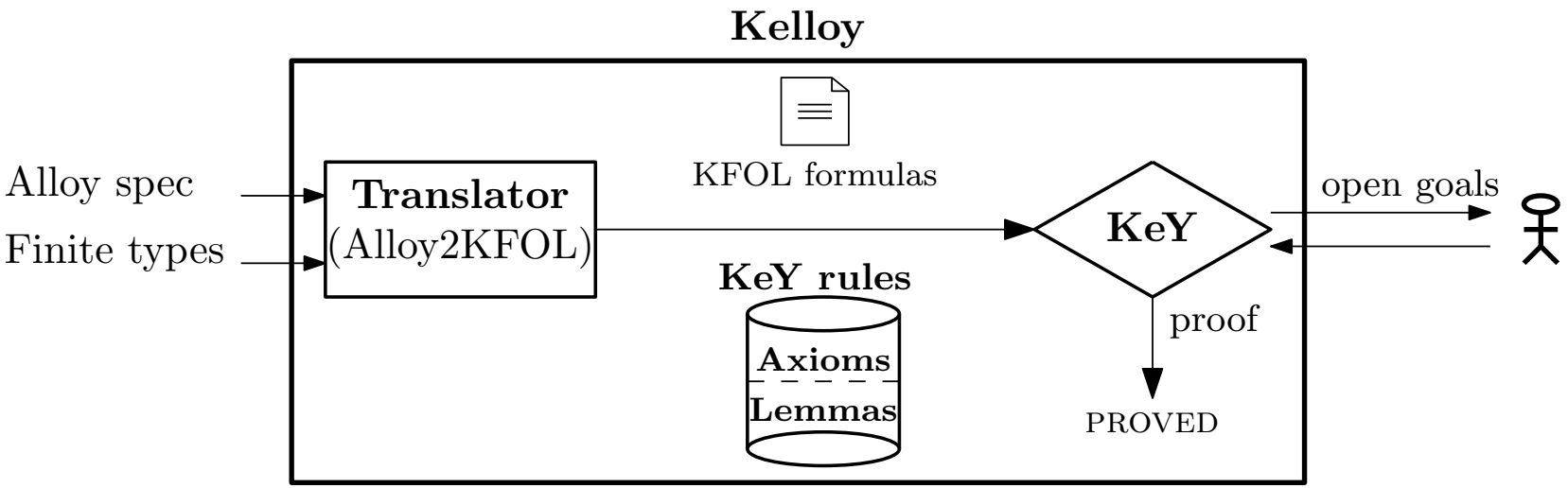
- Performs bounded analysis
  - Requires a user-provided scope
  - Reduction to a satisfiability problem (SAT)
  - Enables automation



- Shortcomings
  - Can never prove an assertion correct, even for the simplest models
  - Limited support for numerical expressions

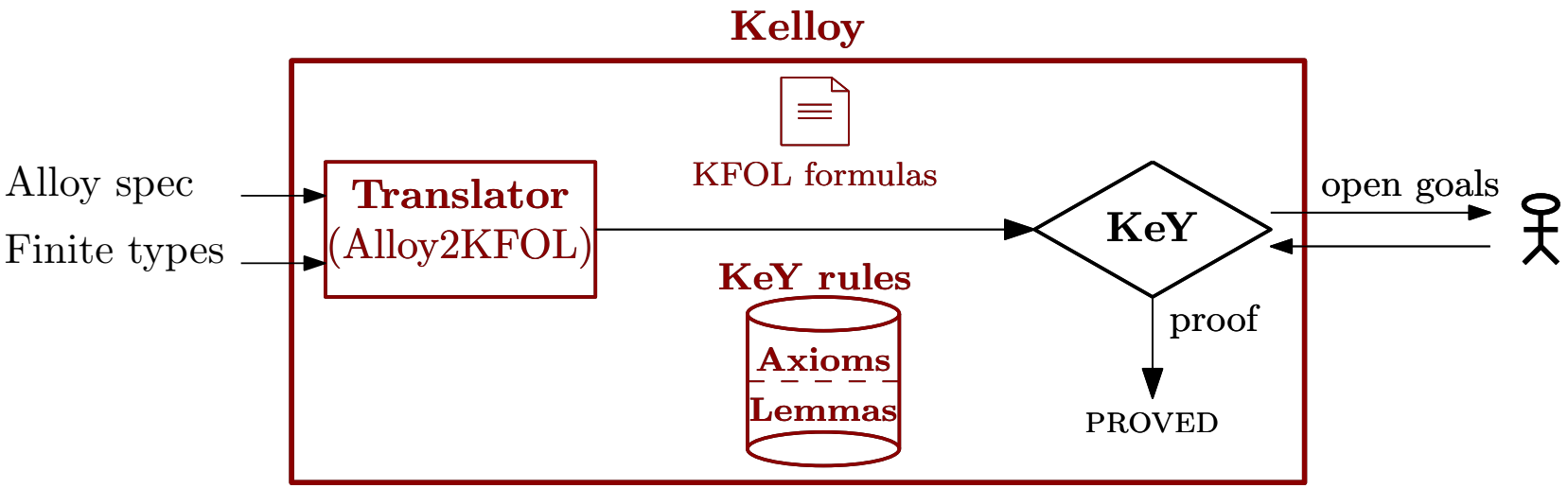
▷ Need for an automatic, proof-capable engine!

Verification – *Approach*



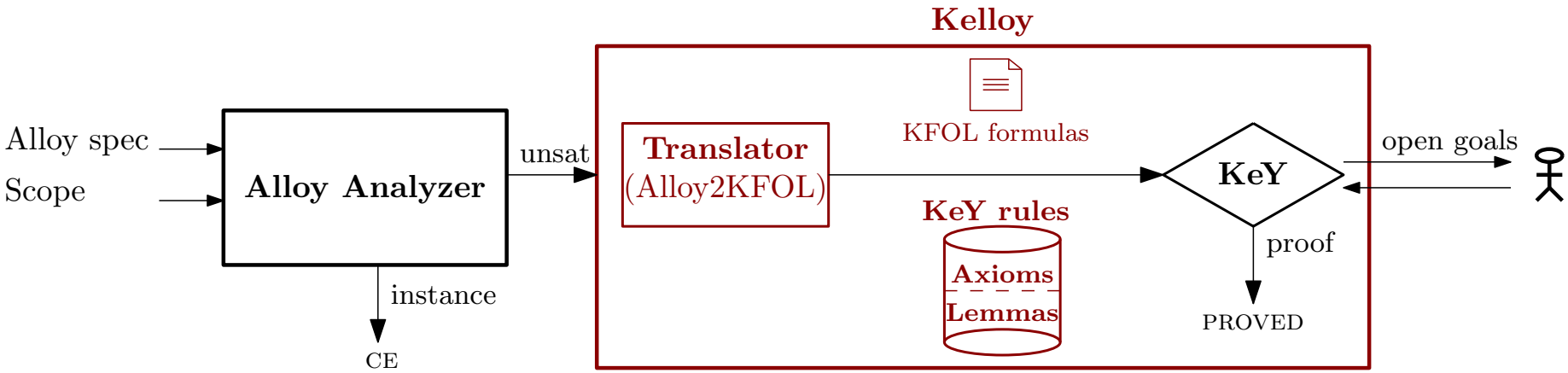
- KeY is as interactive proof engine
  - FOL + integers – *supports Alloy expressiveness*
  - Semi-decidable – *promises higher automation*

Verification – *Approach*



- KeY is as interactive proof engine
  - FOL + integers – *supports Alloy expressiveness*
  - Semi-decidable – *promises higher automation*

Verification – *Approach*



- KeY is as interactive proof engine
  - FOL + integers – *supports Alloy expressiveness*
  - Semi-decidable – *promises higher automation*

Proof Obligation

- Translation

$$T[<P,\ \ fin>] = <Ax, C, k_d, k_a>$$

P	:	An Alloy problem
fin	:	Subset of P's types marked finite
Ax	:	KFOL axioms for the operational semantics
C	:	KFOL constant declarations (signatures and relations)
k_d	:	KFOL formula for declaration constraints and finite types
k_a	:	KFOL formula that encodes the assertion (facts inclusive)

- Corresponding proof obligation:

$$Ax \models_C \mathbf{k_d} \Rightarrow \mathbf{k_a}$$

Translation to KFOL

- Type system: *Rel* and *Atoms*

  Relations becomes first-order structures
- Relational arity is captured by subtyping

  $Rel_1, Rel_2, Rel_3 \ldots$ **extends** *Rel*
- Relational operators as KeY functions
- Semantics by axioms (KeY rules)
- Benefits:
  - Increases the automation level (lemma DB)
  - Simplifies interaction (especially for Alloy users)
- Drawbacks:
  - Operators are arity specific
  - $join_{2\times3} : Rel_2 \times Rel_3 \rightarrow Rel_3$

Example

```
1  abstract sig FSO {
2   parent: lone Dir
3  }
4  sig Dir extends FSO {
5   contents: set FSO
6  }
7  sig File extends FSO {}
8
9  fact {
10  contents = ~parent
11 }
12
13 assert{Φ}
```

```
\functions {
  FSO, Dir, File: Rel1;
  parent, contents: Rel2;}
```

Decls

$\mathbf{subset}_1(\text{Dir}, \text{FSO}),$
$\mathbf{subset}_1(\text{File}, \text{FSO}),$
$\mathbf{disj}_1(\text{Dir}, \text{File}),$
$\mathbf{subset}_1(\text{FSO}, \mathbf{union}_1(\text{Dir}, \text{File}))$

$\mathbf{subset}_2(\text{parent}, \mathbf{prod}_{1\times1}(\text{FSO}, \text{Dir})),$
$\mathbf{subset}_2(\text{contents}, \mathbf{prod}_{1\times1}(\text{Dir}, \text{FSO})),$
$\forall$ this: Atom;
$(\mathbf{in}_1(\text{this}, \text{FSO}) \Rightarrow \mathbf{lone}_1(\mathbf{join}_{1\times2}(\mathbf{sin}(\text{this}), \text{parent}))),$

$\text{contents} = \mathbf{transp}_2(\text{parent})$
$\vdash$
$[\![\Phi]\!]$

Example

```
1  abstract sig FSO {
2   parent: lone Dir
3  }
4  sig Dir extends FSO {
5   contents: set FSO
6  }
7  sig File extends FSO {}
8
9  fact {
10  contents = ~parent
11 }
12
13 assert{Φ}
```

\functions {
FSO, Dir, File: Rel_1;
parent, contents: Rel_2;}

Decls

$\mathbf{subset}_1$(Dir, FSO),
$\mathbf{subset}_1$(File, FSO),
$\mathbf{disj}_1$(Dir, File),
$\mathbf{subset}_1$(FSO, $\mathbf{union}_1$(Dir, File))

$\mathbf{subset}_2$(parent, $\mathbf{prod}_{1\times1}$(FSO, Dir)),
$\mathbf{subset}_2$(contents, $\mathbf{prod}_{1\times1}$(Dir, FSO)),
$\forall$ this: Atom;
($\mathbf{in}_1$(this, FSO) $\Rightarrow$ $\mathbf{lone}_1$($\mathbf{join}_{1\times2}$($\mathbf{sin}$(this), parent))),

contents = $\mathbf{transp}_2$(parent)
$\vdash$
$[\![\Phi]\!]$

Example

```
1  abstract sig FSO {
2   parent: lone Dir
3  }
4  sig Dir extends FSO {
5   contents: set FSO
6  }
7  sig File extends FSO {}
8
9  fact {
10  contents = ~parent
11 }
12
13 assert{Φ}
```

```
\functions {
  FSO, Dir, File: Rel1;
  parent, contents: Rel2;}
```

Decls

$\mathbf{subset}_1(\text{Dir}, \text{FSO}),$
$\mathbf{subset}_1(\text{File}, \text{FSO}),$
$\mathbf{disj}_1(\text{Dir}, \text{File}),$
$\mathbf{subset}_1(\text{FSO}, \mathbf{union}_1(\text{Dir}, \text{File}))$

$\mathbf{subset}_2(\text{parent}, \mathbf{prod}_{1\times1}(\text{FSO}, \text{Dir})),$
$\mathbf{subset}_2(\text{contents}, \mathbf{prod}_{1\times1}(\text{Dir}, \text{FSO})),$
$\forall$ this: Atom;
$(\mathbf{in}_1(\text{this}, \text{FSO}) \Rightarrow \mathbf{lone}_1(\mathbf{join}_{1\times2}(\mathbf{sin}(\text{this}), \text{parent}))),$

$\text{contents} = \mathbf{transp}_2(\text{parent})$
$\vdash$
$[\![\Phi]\!]$

Example

```
1  abstract sig FSO {
2   parent: lone Dir
3  }
4  sig Dir extends FSO {
5   contents: set FSO
6  }
7  sig File extends FSO {}
8
9  fact {
10  contents = ~parent
11 }
12
13 assert{Φ}
```

```
\functions {
  FSO, Dir, File: Rel1;
  parent, contents: Rel2;}
```

Decls

$\mathbf{subset}_1(\text{Dir}, \text{FSO}),$
$\mathbf{subset}_1(\text{File}, \text{FSO}),$
$\mathbf{disj}_1(\text{Dir}, \text{File}),$
$\mathbf{subset}_1(\text{FSO}, \mathbf{union}_1(\text{Dir}, \text{File}))$

$\mathbf{subset}_2(\text{parent}, \mathbf{prod}_{1\times1}(\text{FSO}, \text{Dir})),$
$\mathbf{subset}_2(\text{contents}, \mathbf{prod}_{1\times1}(\text{Dir}, \text{FSO})),$
$\forall$ this: Atom;
$(\mathbf{in}_1(\text{this}, \text{FSO}) \Rightarrow \mathbf{lone}_1(\mathbf{join}_{1\times2}(\mathbf{sin}(\text{this}), \text{parent}))),$

$\text{contents} = \mathbf{transp}_2(\text{parent})$
$\vdash$
$[\![\Phi]\!]$

Example

```
1  abstract sig FSO {
2   parent: lone Dir
3  }
4  sig Dir extends FSO {
5   contents: set FSO
6  }
7  sig File extends FSO {}
8
9  fact {
10  contents = ~parent
11 }
12
13 assert{Φ}
```

```
\functions {
  FSO, Dir, File: Rel1;
  parent, contents: Rel2;}
```

Decls

$\mathbf{subset}_1(\text{Dir}, \text{FSO}),$
$\mathbf{subset}_1(\text{File}, \text{FSO}),$
$\mathbf{disj}_1(\text{Dir}, \text{File}),$
$\mathbf{subset}_1(\text{FSO}, \mathbf{union}_1(\text{Dir}, \text{File}))$

$\mathbf{subset}_2(\text{parent}, \mathbf{prod}_{1\times1}(\text{FSO}, \text{Dir})),$
$\mathbf{subset}_2(\text{contents}, \mathbf{prod}_{1\times1}(\text{Dir}, \text{FSO})),$
$\forall$ this: Atom;
$(\mathbf{in}_1(\text{this}, \text{FSO}) \Rightarrow \mathbf{lone}_1(\mathbf{join}_{1\times2}(\mathbf{sin}(\text{this}), \text{parent}))),$

$\text{contents} = \mathbf{transp}_2(\text{parent})$
$\vdash$
$[\![\Phi]\!]$

Example

```
1  abstract sig FSO {
2   parent: lone Dir
3  }
4  sig Dir extends FSO {
5   contents: set FSO
6  }
7  sig File extends FSO {}
8
9  fact {
10  contents = ~parent
11 }
12
13 assert{Φ}
```

```
\functions {
  FSO, Dir, File: Rel1;
  parent, contents: Rel2;}
```

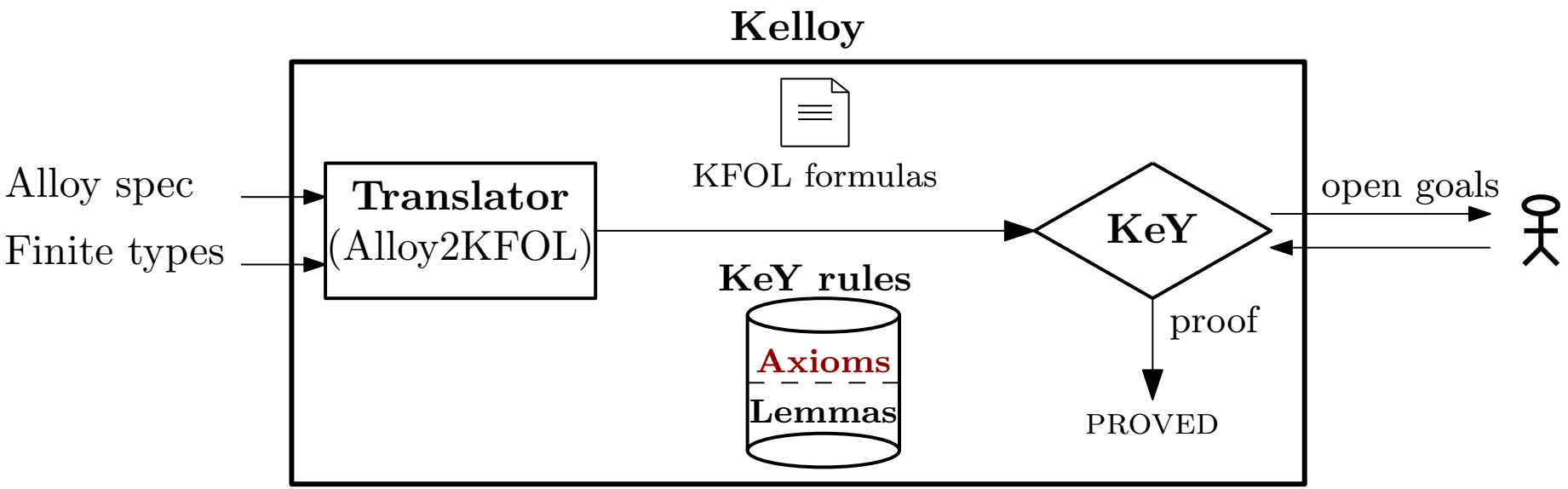
Decls

$$\mathbf{subset}_1(\text{Dir}, \text{FSO}),$$
$$\mathbf{subset}_1(\text{File}, \text{FSO}),$$
$$\mathbf{disj}_1(\text{Dir}, \text{File}),$$
$$\mathbf{subset}_1(\text{FSO}, \mathbf{union}_1(\text{Dir}, \text{File}))$$

$$\mathbf{subset}_2(\text{parent}, \mathbf{prod}_{1\times1}(\text{FSO}, \text{Dir})),$$
$$\mathbf{subset}_2(\text{contents}, \mathbf{prod}_{1\times1}(\text{Dir}, \text{FSO})),$$
$$\forall\ \text{this: Atom};$$
$$(\mathbf{in}_1(\text{this}, \text{FSO}) \Rightarrow \mathbf{lone}_1(\mathbf{join}_{1\times2}(\mathbf{sin}(\text{this}), \text{parent}))),$$

$$\text{contents} = \mathbf{transp}_2(\text{parent})$$
$$\vdash$$
$$[\![\Phi]\!]$$

Relational Theory Axiomatization



Relational Theory Axiomatization

- Membership predicates for the basic set theoretical semantics:
  $in_n \subseteq \underbrace{\text{Atom} \times \cdots \times \text{Atom}}_{n \text{ times}} \times \text{Rel}_n$
- Axioms for the operational semantics, e.g.

$$\forall r\colon Rel_1, s\colon Rel_2, a\colon Atom \mid$$
$$in_1(a, join_{1\times 2}(r, s)) \Leftrightarrow (\exists b\colon Atom \mid in_1(b, r) \wedge in_2(b, a, s))$$

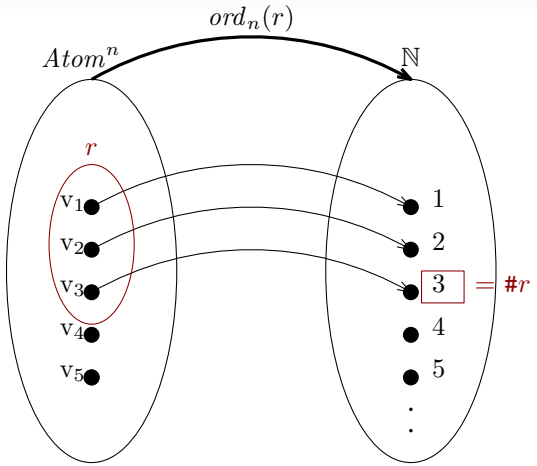
- Axioms are implemented as KeY rules, e.g.

```
joinDef1x2 {
  \find (in(a, join1x2(r, s)))
  \replacewith (∃ b; (in1(b, r) ∧ in2(b, a, s)))
};
```

Axioms

Cardinality

$\#r$ is defined only if r is finite



$$\forall r\colon Rel_n, a_{1:n}\colon Atom \mid (\mathbf{finite}_n(r) \wedge in_n(a_{1:n}, r)) \\ \Rightarrow 1 \leq ord_n(r, a_{1:n}) \leq card_n(r)$$

$$\forall r\colon Rel_n, i\colon int \mid \mathbf{finite}_n(r) \wedge 1 \leq i \leq card_n(r)) \\ \Rightarrow \exists a_{1:n}\colon Atom \mid in_n(a_{1:n}, r) \wedge ord_n(r, a_{1:n}) = i$$

$$\forall r\colon Rel_n, a_{1:n}, b_{1:n}\colon Atom \mid \\ (\mathbf{finite}_n(r) \wedge in_n(a_{1:n}, r) \wedge in_n(b_{1:n}, r) \wedge ord_n(r, a_{1:n}) = ord_n(r, b_{1:n})) \\ \Rightarrow (a_1 = b_1 \wedge \ldots \wedge a_n = b_n)$$

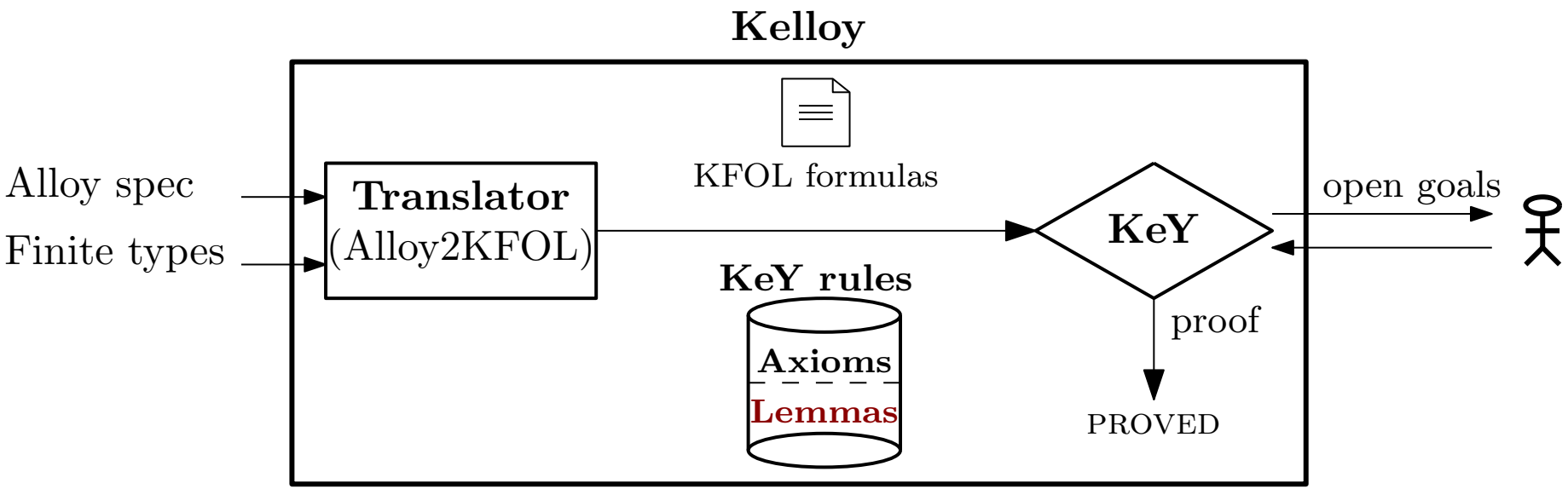
Transitive Closure

- Alloy semantics: $\hat{}\mathtt{r} = \mathtt{r} \cup \mathtt{r.r} + \cup \ldots \cup \mathtt{r}^{(n)}$
- $T[\hat{}\mathtt{r}] = tc_2(r)$, axiomatized using KeY integers

$$\forall r\colon Rel_2, a, b\colon Atom \mid$$
$$in_2(a, b, tc_2(r)) \Leftrightarrow \exists i\colon int \mid i \geq 0 \wedge in_2(a, b, itrJoin_2(r, i))$$

$$\forall r\colon Rel_2, i\colon int^{\geq 0} \mid$$
$$itrJoin_2(r, 0) = r \wedge$$
$$itrJoin_2(r, i+1) = union_2(itrJoin_2(r, i), join_{2\times 2}(r, itrJoin_2(r, i))))$$

Proving Strategy for efficient relational reasoning



Proving Strategy for efficient relational reasoning

- Using semantics axioms:

$$\frac{\Gamma \cup \{(\forall x_{1:n} : Atom^n \mid in_n(x_{1:n}, r) \Rightarrow in_n(x_{1:n}, s)), in_n(a_{1:n}, r)\} \vdash \Delta}{\Gamma \cup \{subset_n(r, s), in_n(a_{1:n}, r)\} \vdash \Delta}$$

- Using lemmas:

$$\frac{\Gamma \cup \{subset_n(r, s), in_n(a_{1:n}, r), in_n(a_{1:n}, s)\} \vdash \Delta}{\Gamma \cup \{subset_n(r, s), in_n(a_{1:n}, r)\} \vdash \Delta}$$

Keep predicate invocations in the antecedent; use lemmas to exploit their semantics

Proving Strategy for efficient relational reasoning

- Using semantics axioms:

$$\frac{\Gamma \cup \{(\forall x_{1:n} : Atom^n \mid in_n(x_{1:n}, r) \Rightarrow in_n(x_{1:n}, s)), in_n(a_{1:n}, r)\} \vdash \Delta}{\Gamma \cup \{subset_n(r, s), in_n(a_{1:n}, r)\} \vdash \Delta}$$

- Using lemmas:

$$\frac{\Gamma \cup \{subset_n(r, s), in_n(a_{1:n}, r), in_n(a_{1:n}, s)\} \vdash \Delta}{\Gamma \cup \{subset_n(r, s), in_n(a_{1:n}, r)\} \vdash \Delta}$$

Keep predicate invocations in the antecedent; use lemmas to exploit their semantics

Proving Strategy for efficient relational reasoning

- Using semantics axioms:

$$\frac{\Gamma \cup \{(\forall x_{1:n} : Atom^n \mid in_n(x_{1:n}, r) \Rightarrow in_n(x_{1:n}, s)), in_n(a_{1:n}, r)\} \vdash \Delta}{\Gamma \cup \{subset_n(r, s), in_n(a_{1:n}, r)\} \vdash \Delta}$$

- Using lemmas:

$$\frac{\Gamma \cup \{subset_n(r, s), in_n(a_{1:n}, r), in_n(a_{1:n}, s)\} \vdash \Delta}{\Gamma \cup \{subset_n(r, s), in_n(a_{1:n}, r)\} \vdash \Delta}$$

Keep predicate invocations in the antecedent; use lemmas to exploit their semantics

Proving Strategy for efficient relational reasoning

- Using semantics axioms:

$$\frac{\Gamma \cup \{(\forall x_{1:n} : Atom^n \mid in_n(x_{1:n}, r) \Rightarrow in_n(x_{1:n}, s)), in_n(a_{1:n}, r)\} \vdash \Delta}{\Gamma \cup \{subset_n(r, s), in_n(a_{1:n}, r)\} \vdash \Delta}$$

- Using lemmas:

$$\frac{\Gamma \cup \{subset_n(r, s), in_n(a_{1:n}, r), in_n(a_{1:n}, s)\} \vdash \Delta}{\Gamma \cup \{subset_n(r, s), in_n(a_{1:n}, r)\} \vdash \Delta}$$

Keep predicate invocations in the antecedent; use lemmas to exploit their semantics

Proving Strategy for efficient relational reasoning

- Set theoretical lemmas:
  - Capture useful and frequently needed property
  - Avoid prover's quantifier instantiation strategy
- Lemmas for transitive closure:
  - More important – *can avoid induction*
  - Shorten proofs, e.g.

$$\frac{\Gamma \cup \{subset_2(r_2, s_2), subset_2(transClos(r_2), transClos(s_2))\} \vdash \Delta}{\Gamma \cup \{subset_2(r_2, s_2)\} \vdash \Delta}$$

- Simplification lemmas are applied greedily, e.g.

$$union_n(r_n, r_n) \rightsquigarrow r_n \quad , \quad \frac{subset_n(r_n, s_n)}{union_n(r_n, s_n) \rightsquigarrow s_n}$$

- $\sim$ 100 provided lemmas are proved
- Benefit confirmed by experiments

Proving Strategy for efficient relational reasoning

- Set theoretical lemmas:
  - Capture useful and frequently needed property
  - Avoid prover's quantifier instantiation strategy
- Lemmas for transitive closure:
  - More important – *can avoid induction*
  - Shorten proofs, e.g.

$$\frac{\Gamma \cup \{subset_2(r_2, s_2), subset_2(transClos(r_2), transClos(s_2))\} \vdash \Delta}{\Gamma \cup \{subset_2(r_2, s_2)\} \vdash \Delta}$$

- Simplification lemmas are applied greedily, e.g.

$$union_n(r_n, r_n) \rightsquigarrow r_n \quad , \quad \frac{subset_n(r_n, s_n)}{union_n(r_n, s_n) \rightsquigarrow s_n}$$

- $\sim$ 100 provided lemmas are proved
- Benefit confirmed by experiments

Proving Strategy for efficient relational reasoning

- Set theoretical lemmas:
  - Capture useful and frequently needed property
  - Avoid prover's quantifier instantiation strategy
- Lemmas for transitive closure:
  - More important – *can avoid induction*
  - Shorten proofs, e.g.

$$\frac{\Gamma \cup \{subset_2(r_2, s_2), subset_2(transClos(r_2), transClos(s_2))\} \vdash \Delta}{\Gamma \cup \{subset_2(r_2, s_2)\} \vdash \Delta}$$

- Simplification lemmas are applied greedily, e.g.

$$union_n(r_n, r_n) \rightsquigarrow r_n \quad , \quad \frac{subset_n(r_n, s_n)}{union_n(r_n, s_n) \rightsquigarrow s_n}$$

- $\sim$ 100 provided lemmas are proved
- Benefit confirmed by experiments

Proving Strategy for efficient relational reasoning

- Set theoretical lemmas:
  - Capture useful and frequently needed property
  - Avoid prover's quantifier instantiation strategy
- Lemmas for transitive closure:
  - More important – *can avoid induction*
  - Shorten proofs, e.g.

$$\frac{\Gamma \cup \{subset_2(r_2, s_2), subset_2(transClos(r_2), transClos(s_2))\} \vdash \Delta}{\Gamma \cup \{subset_2(r_2, s_2)\} \vdash \Delta}$$

- Simplification lemmas are applied greedily, e.g.

$$union_n(r_n, r_n) \rightsquigarrow r_n \quad , \quad \frac{subset_n(r_n, s_n)}{union_n(r_n, s_n) \rightsquigarrow s_n}$$

- $\sim$ 100 provided lemmas are proved
- Benefit confirmed by experiments

Proving Strategy for efficient relational reasoning

- Set theoretical lemmas:
  - Capture useful and frequently needed property
  - Avoid prover's quantifier instantiation strategy
- Lemmas for transitive closure:
  - More important – *can avoid induction*
  - Shorten proofs, e.g.

$$\frac{\Gamma \cup \{subset_2(r_2, s_2), subset_2(transClos(r_2), transClos(s_2))\} \vdash \Delta}{\Gamma \cup \{subset_2(r_2, s_2)\} \vdash \Delta}$$

- Simplification lemmas are applied greedily, e.g.

$$union_n(r_n, r_n) \rightsquigarrow r_n \quad , \quad \frac{subset_n(r_n, s_n)}{union_n(r_n, s_n) \rightsquigarrow s_n}$$

- $\sim$ 100 provided lemmas are proved
- Benefit confirmed by experiments

Evaluation – *Alloy specific strategy*

		Kelloy Strategy		Basic Strategy	
Problem	Assertion	Time (Steps)	Result	Time (Steps)	Result
address book	delUndoesAdd	9.3 (2476)	proved	27.1 (5475)	proved
	addIdempotent	0.1 (113)	proved	5.0 (1176)	proved
abstract memory	writeRead	0.8 (567)	proved	1.0 (597)	proved
	writeIdempotent	14.0 (4482)	proved	6.5 (1009)	proved
media assets	hidePreservesInv	0.0 (39)	proved	0.1 (70)	proved
	pasteNotAffectsHidden	15.9 (2619)	proved	time-out (–)	failed
mark sweep	soundness1	3.0 (1195)	proved	time-out (–)	failed
grandpa	noSelfFather	0.0 (77)	proved	0.0 (77)	proved
	noSelfGrandpa	26.5 (3144)	proved	39.8 (3276)	proved
filesystem	FileInDir	0.5 (160)	proved	time-out (–)	failed
	SomeDir	0.2 (205)	proved	time-out (–)	failed
birthday	addWorks	0.1 (129)	proved	1.2 (506)	proved

(time in seconds, time-out after 2h)

- Out of 22 assertions 12 have been proved fully automatically
- Improvement over the basic strategy in 11 cases out of 12

Evaluation – *Alloy specific strategy*

		Kelloy Strategy		Basic Strategy	
Problem	Assertion	Time (Steps)	Result	Time (Steps)	Result
address book	delUndoesAdd	9.3 (2476)	proved	27.1 (5475)	proved
	addIdempotent	0.1 (113)	proved	5.0 (1176)	proved
abstract memory	writeRead	0.8 (567)	proved	1.0 (597)	proved
	writeIdempotent	14.0 (4482)	proved	6.5 (1009)	proved
media assets	hidePreservesInv	0.0 (39)	proved	0.1 (70)	proved
	pasteNotAffectsHidden	15.9 (2619)	proved	time-out (–)	failed
mark sweep	soundness1	3.0 (1195)	proved	time-out (–)	failed
grandpa	noSelfFather	0.0 (77)	proved	0.0 (77)	proved
	noSelfGrandpa	26.5 (3144)	proved	39.8 (3276)	proved
filesystem	FileInDir	0.5 (160)	proved	time-out (–)	failed
	SomeDir	0.2 (205)	proved	time-out (–)	failed
birthday	addWorks	0.1 (129)	proved	1.2 (506)	proved

(time in seconds, time-out after 2h)

- Out of 22 assertions 12 have been proved fully automatically
- Improvement over the basic strategy in 11 cases out of 12

Evaluation – *interactive proofs (done by expert)*

- 10 of the verified assertions required user interactions interactions
- Three groups of user interaction (1)/(2)/(3)
  - (1) Hypothesis introduction, e.g. *induction hypothesis*
  - (2) Prover guidance, e.g. *quantifier instantiations*
  - (3) Non-essential steps
- 7 assertions required less than 10 interactions

  e.g. *completeness* assertion of mark-and-sweep (1/0/0)
- The remaining 3 assertions required between 36 to 291 interactions
- Most complex proof: *correctness of Dijkstra's deadlock prevention*
  - 18875 steps (overall)
  - 291 (7/219/65) interactions

Evaluation – *effect of user's expertise*

Proof of the *soundness2* assertion for mark-and-sweep

- By an Alloy user with no previous experience in KeY
  - 1389 steps (overall)
  - 207 (2/57/148) interactions
- By an experienced user in both Alloy and KeY
  - 9372 steps (overall)
  - 10 (5/1/4) interactions

Related work

- Prioni [Arkoudas, Khurshid, Marinov, Rinard, 2003] uses Athena
  - Similar to ours, translates Alloy operators to FOL functions
  - Cannot handle infinite types
- Dynamite [Frias, Pombo, Moscato, 2007] uses PVS
  - Translates Alloy to omega closure fork algebras
  - Targets a higher-order logic
- Prioni and Dynamite:
  - No results about automation level
  - No support of integer or cardinality expressions
  - (Dynamite) Reduction to binary relations results in additional proof obligations
- SMT-based engine [El Ghazi, Taghdiri, 2011] uses Z3
  - Fully automatic proofs
  - Support of integer and cardinality expressions
  - No completeness guarantee

Conclusions

- A tool for full verification of Alloy problems
  - Supports the whole Alloy language
  - Preserves same problem structure as Alloy

    $\Rightarrow$ For automation and readability
  - Implements a relational theory in KeY
  - Provides a lemma database
  - Extends KeY reasoning strategy
- Discussion of correctness and completeness properties (see paper)
- Evaluation of the approach and the extended strategy
  - We proved a total for 22 assertions in 10 Alloy problems
  - 12 of them have been proved completely automatically
  - Lemmas increase the automation level considerably
  - Usually, only structurally complex systems or systems involving inductive properties require user interaction