JKelloy: A Proof Assistant for Relational Specifications of Java Programs

Aboubakr Achraf El Ghazi, Mattias Ulbrich, Christoph Gladisch,
Shmuel Tyszberowicz, Mana Taghdiri

Karlsruhe Institute of Technology, Germany

NFM 2014
Houston, April 30

Bringing together

Java + ALLOY

- Classes
- Fields
- Linked Data Structures
- Program states

- Sets
- Relations
- First-Order Relational Logic
- No states

Relational Specifications are Concise

```
class Tree {
  Tree left, right;
  Data data;
}
```

$$\rightsquigarrow \quad \begin{aligned} \mathit{Tree}, \mathit{Data} &\subseteq \mathit{Object} \\ \mathit{left}, \mathit{right} &\subseteq \mathit{Tree} \times \mathit{Tree} \cup \mathit{Null} \\ \mathit{data} &\subseteq \mathit{Tree} \times \mathit{Data} \cup \mathit{Null} \end{aligned}$$

Relational Specifications are Concise

```
class Tree {
  Tree left, right;
  Data data;
}
```

$\rightsquigarrow$

$$\begin{aligned} \mathit{Tree}, \mathit{Data} &\subseteq \mathit{Object} \\ \mathit{left}, \mathit{right} &\subseteq \mathit{Tree} \times \mathit{Tree} \cup \mathit{Null} \\ \mathit{data} &\subseteq \mathit{Tree} \times \mathit{Data} \cup \mathit{Null} \end{aligned}$$

- All instances of `Tree` are acyclic:

$$\forall t : \mathit{Tree} \mid t \notin t\,\texttt{.}\,\hat{}\,(\mathit{left}\ \texttt{+}\ \mathit{right})$$

Relational Specifications are Concise

```
class Tree {
  Tree left, right;
  Data data;
}
```

$\rightsquigarrow$

$$Tree, Data \subseteq Object$$
$$left, right \subseteq Tree \times Tree \cup Null$$
$$data \subseteq Tree \times Data \cup Null$$

- All instances of `Tree` are acyclic:

$$\forall t : Tree \mid t \notin t\,.\,\hat{}\,(left\ +\ right)$$

- A tree t is a subtree of the tree `root`:

$$t \in root\,.\,*(left\ +\ right)$$

Relational Specifications are Concise

```
class Tree {
  Tree left, right;
  Data data;
}
```

$\rightsquigarrow$

$$Tree, Data \subseteq Object$$
$$left, right \subseteq Tree \times Tree \cup Null$$
$$data \subseteq Tree \times Data \cup Null$$

- All instances of `Tree` are acyclic:

$$\forall t : Tree \mid t \notin t\,.\,\hat{}\,(left\ +\ right)$$

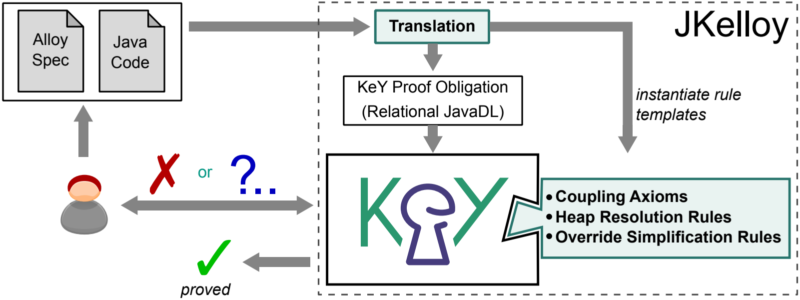
- A tree t is a subtree of the tree `root`:

$$t \in root\,.\,{*}(left\ +\ right)$$

- The data d occurs at most once in the tree `root`:

$$\#((root\,.\,{*}(left\ +\ right))\ \&\ (data\,.\,d)) \leq 1$$

JKelloy – *Framework*



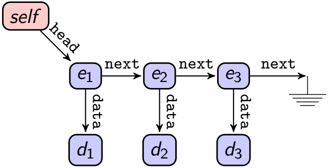
- Alloy as specification language
- KeY for symbolic execution
- Heap resolution rules: Translate Java state to Alloy
- Override resolution rules: Simplify relational expressions

Prepend Example

```
class List {
 Entry head;

 void prepend(Object d) {
   Entry oldHead = head;
   head = new Entry();
   head.next = oldHead;
   head.data = d;
 }
}

class Entry {
 Data data;
 Entry next;
}
```

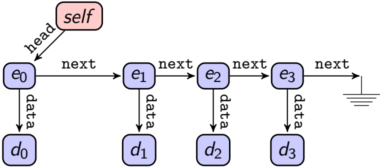


Prepend Example

```
class List {
 Entry head;

 void prepend(Object d) {
   Entry oldHead = head;
   head = new Entry();
   head.next = oldHead;
   head.data = d;
 }
}

class Entry {
 Data data;
 Entry next;
}
```



Construction of Proof Obligations

$$pre \rightarrow [\texttt{prepend(d);}]post$$

symbolic execution

$$pre \wedge path_1 \rightarrow \{\mathcal{U}_1\}post \quad \cdots \quad pre \wedge path_n \rightarrow \{\mathcal{U}_n\}post$$

Construction of Proof Obligations

$$pre \rightarrow \{h_s := heap\}[\texttt{prepend(d);}]\{h_e := heap\}post$$

symbolic execution

$$pre \wedge path_1 \rightarrow \{\mathcal{U}_1\}post \quad \cdots \quad pre \wedge path_n \rightarrow \{\mathcal{U}_n\}post$$

Construction of Proof Obligations

$$pre \rightarrow \{h_s := heap\}[\texttt{prepend(d);}]\{h_e := heap\}post$$

symbolic execution

$$pre \wedge path_1 \rightarrow \{\mathcal{U}_1\}post \quad \cdots \quad pre \wedge path_n \rightarrow \{\mathcal{U}_n\}post$$

h_s

Construction of Proof Obligations

$$pre \;\rightarrow\; \{h_s := heap\}[\texttt{prepend(d);}]\{h_e := heap\}post$$

symbolic execution

$$pre \wedge path_1 \rightarrow \{\mathcal{U}_1\}post \quad \cdots \quad pre \wedge path_n \rightarrow \{\mathcal{U}_n\}post$$

h_s

$\downarrow$

$head = newEntry() \rightsquigarrow \qquad create(h_s, e)$

Construction of Proof Obligations

$$pre \rightarrow \{h_s := heap\}[\texttt{prepend(d);}]\{h_e := heap\}post$$

symbolic execution

$$pre \wedge path_1 \rightarrow \{\mathcal{U}_1\}post \quad \cdots \quad pre \wedge path_n \rightarrow \{\mathcal{U}_n\}post$$

$head = newEntry() \rightsquigarrow$

$$h_s$$
$$\downarrow$$
$$create(h_s, e)$$
$$\downarrow$$
$$store(create(h_s, e), self, \texttt{head}, e)$$

Construction of Proof Obligations

$$pre \rightarrow \{h_s := heap\}[\texttt{prepend(d);}]\{h_e := heap\}post$$

symbolic execution

$$pre \wedge path_1 \rightarrow \{\mathcal{U}_1\}post \quad \cdots \quad pre \wedge path_n \rightarrow \{\mathcal{U}_n\}post$$

$$h_s$$

$$\downarrow$$

$head = newEntry() \rightsquigarrow$

$$create(h_s, e)$$

$$\downarrow$$

$$store(create(h_s, e), self, \texttt{head}, e)$$

$$\downarrow$$

$head.next = oldHead \rightsquigarrow$

$$store(store(create(h_s, e), self, \texttt{head}, e), e, \texttt{next}, h_s[self.\texttt{head}])$$

Construction of Proof Obligations

$$pre \;\rightarrow\; \{h_s := heap\}[\texttt{prepend(d);}]\{h_e := heap\}post$$

symbolic execution

$$pre \;\wedge\; path_1 \;\rightarrow\; \{\mathcal{U}_1\}post \quad \cdots \quad pre \;\wedge\; path_n \;\rightarrow\; \{\mathcal{U}_n\}post$$

$$h_s$$

$$\downarrow$$

$head = newEntry() \rightsquigarrow$

$$create(h_s, e)$$

$$\downarrow$$

$$store(create(h_s, e), self, \texttt{head}, e)$$

$$\downarrow$$

$head.next = oldHead \rightsquigarrow$

$$store(store(create(h_s, e), self, \texttt{head}, e), e, \texttt{next}, h_s[self.\texttt{head}])$$

$$\downarrow$$

$head.data = d \rightsquigarrow$

$$store(store(store(create(h_s, e), self, \texttt{head}, e), e, \texttt{next}, h_s[self.\texttt{head}]), e, \texttt{data}, d)$$

Coupling Alloy and Java States

```
class List {
 Entry head;
 void prepend(Data d) {...}
}

class Entry {
 Data data;
 Entry next;
}
```

$$\begin{aligned} \mathit{List}, \mathit{Entry}, \mathit{Data} &\subseteq \mathit{Object} \\ \mathit{head} &\subseteq \mathit{List} \times \mathit{Entry} \cup \mathit{Null} \\ \mathit{data} &\subseteq \mathit{Entry} \times \mathit{Data} \cup \mathit{Null} \\ \mathit{next} &\subseteq \mathit{Entry} \times \mathit{Entry} \cup \mathit{Null} \end{aligned}$$

Coupling Axioms in JavaDL (KeY 's logic)

$$\mathit{Entry}_{rel}(h) := \{o \mid h[o.\langle\texttt{created}\rangle] \ \wedge \ o \sqsubseteq \mathit{Entry} \ \wedge \ o \neq \texttt{null}\}$$

$$\mathit{data}_{rel}(h) := \{(o_1, o_2) \mid o_1 \in \mathit{Entry}_{rel}(h) \ \wedge \ (o_2 = \texttt{null} \vee o_2 \in \mathit{Data}_{rel}(h)) \ \wedge \ o_2 = h[o_1.\texttt{data}]\}$$

$$\mathit{Null}_{rel}(h) := \{\texttt{null}\}$$

Coupling Alloy and Java States

```
class List {
 Entry head;
 void prepend(Data d) {...}
}

class Entry {
 Data data;
 Entry next;
}
```

$$
\begin{aligned}
List, Entry, Data &\subseteq Object \\
head &\subseteq List \times Entry \cup Null \\
data &\subseteq Entry \times Data \cup Null \\
next &\subseteq Entry \times Entry \cup Null
\end{aligned}
$$

Coupling Axioms in JavaDL (KeY 's logic)

$$Entry_{rel}(h) := \{o \mid h[o.\langle\mathtt{created}\rangle] \ \wedge \ o \sqsubseteq Entry \ \wedge \ o \neq \mathtt{null}\}$$

$$data_{rel}(h) := \{(o_1, o_2) \mid o_1 \in Entry_{rel}(h) \ \wedge \ (o_2 = \mathtt{null} \vee o_2 \in Data_{rel}(h)) \ \wedge \ o_2 = h[o_1.\mathtt{data}]\}$$

$$Null_{rel}(h) := \{\mathtt{null}\}$$

Coupling Alloy and Java States

```
class List {
 Entry head;
 void prepend(Data d) {...}
}

class Entry {
 Data data;
 Entry next;
}
```

$$\begin{array}{ll} \multicolumn{2}{c}{List, Entry, Data \subseteq Object} \\ head & \subseteq List \times Entry \cup Null \\ data & \subseteq Entry \times Data \cup Null \\ next & \subseteq Entry \times Entry \cup Null \end{array}$$

Coupling Axioms in JavaDL (KeY 's logic)

$$Entry_{rel}(h) := \{o \mid h[o.\langle\texttt{created}\rangle] \ \wedge \ o \sqsubseteq Entry \ \wedge \ o \neq \texttt{null}\}$$

$$data_{rel}(h) := \{(o_1, o_2) \mid o_1 \in Entry_{rel}(h) \ \wedge \ (o_2 = \texttt{null} \vee o_2 \in Data_{rel}(h)) \ \wedge \ o_2 = h[o_1.\texttt{data}]\}$$

$$Null_{rel}(h) := \{\texttt{null}\}$$

Coupling Alloy and Java States

```
class List {
 Entry head;
 void prepend(Data d) {...}
}

class Entry {
 Data data;
 Entry next;
}
```

$$
\begin{aligned}
&List, Entry, Data \subseteq Object \\
&head \quad \subseteq List \times Entry \cup Null \\
&data \quad \subseteq Entry \times Data \cup Null \\
&next \quad \subseteq Entry \times Entry \cup Null
\end{aligned}
$$

Coupling Axioms in JavaDL (KeY 's logic)

$$Entry_{rel}(h) := \{o \mid h[o.\langle\mathtt{created}\rangle] \ \wedge \ o \sqsubseteq Entry \ \wedge \ o \neq \mathtt{null}\}$$

$$data_{rel}(h) := \{(o_1, o_2) \mid o_1 \in Entry_{rel}(h) \ \wedge \ (o_2 = \mathtt{null} \vee o_2 \in Data_{rel}(h)) \ \wedge \ o_2 = h[o_1.\mathtt{data}]\}$$

$$Null_{rel}(h) := \{\mathtt{null}\}$$

Relational Heap Resolution – *Motivation*

Evaluating an Alloy relation in a Java heap context

$$a\,.\,f(store(h_i, o, g, v)) \stackrel{?}{=} b$$

Relational Heap Resolution – *Motivation*

Evaluating an Alloy relation in a Java heap context

$$a\,.\,f(store(h_i, o, g, v)) \stackrel{?}{=} b$$

Coupling Axioms

$a \in C(store(h_i, o, g, v)) \wedge$
$(b = \texttt{null} \vee b \in T(store(h_i, o, g, v))) \wedge$
$b = store(h_i, o, g, v)[a.\texttt{f}]$

Relational Heap Resolution – *Motivation*

Evaluating an Alloy relation in a Java heap context

$$a\,.\,f(store(h_i, o, g, v)) \stackrel{?}{=} b$$

Coupling Axioms

$a \in C(store(h_i, o, g, v)) \wedge$

$(b = \mathtt{null} \vee b \in T(store(h_i, o, g, v))) \wedge$

$b = \underline{store(h_i, o, g, v)[a.\mathtt{f}]}$

Select/Store Axioms

$$store(h_i, o, g, v)[a.\mathtt{f}] = \begin{cases} v & \text{if } o = a \wedge \mathtt{g} = \mathtt{f} \wedge \mathtt{g} \neq \mathtt{created} \\ h_i[a.\mathtt{f}] & \text{else} \end{cases}$$

Relational Heap Resolution – *Motivation*

Evaluating an Alloy relation in a Java heap context

$$a\,.\,f(store(h_i, o, g, v)) \stackrel{?}{=} b$$

Coupling Axioms

$a \in C(store(h_i, o, g, v)) \wedge$
$(b = \texttt{null} \vee b \in T(store(h_i, o, g, v))) \wedge$
$b = store(h_i, o, g, v)[a.\texttt{f}]$

$+$

Select/Store Axioms

$$store(h_i, o, g, v)[a.\texttt{f}] = \begin{cases} v & \textit{if } o = a \wedge \texttt{g} = \texttt{f} \wedge \texttt{g} \neq \texttt{created} \\ h_i[a.\texttt{f}] & \textit{else} \end{cases}$$

→ **Relational Heap Resolution**

Heap Resolution Rules – *Example*

- Postcondition of `prepend(d)` (in Alloy)

$$self\,.\,head'\,.\,next'^{*}\,.\,data' = self\,.\,head\,.\,next^{*}\,.\,data \cup d$$

Heap Resolution Rules – *Example*

- Postcondition of `prepend(d)` (in Alloy)

$$self \,.\, head' \,.\, next'^* \,.\, data' = self \,.\, head \,.\, next^* \,.\, data \cup d$$

- In relational JavaDL (KeY's logic)

$$\{self\} \,.\, head_{rel}(h_e) \,.\, next_{rel}(h_e)^* \,.\, data_{rel}(h_e)$$
$$=\{self\} \,.\, head_{rel}(h_s) \,.\, next_{rel}(h_s)^* \,.\, data_{rel}(h_s) \cup \{d\}$$

$$h_e = store(store(store(create(h_s, e), self, \mathtt{head}, e), e, \mathtt{next}, h_s[self.\mathtt{head}]), e, \mathtt{data}, d)$$

Heap Resolution Rules – *Example*

- Postcondition of `prepend(d)` (in Alloy)

$$self\,.\,head'\,.\,next'^{*}\,.\,data' = self\,.\,head\,.\,next^{*}\,.\,data \cup d$$

- In relational JavaDL (KeY's logic)

$$\{self\}\,.\,head_{rel}(h_e)\,.\,next_{rel}(h_e)^{*}\,.\,data_{rel}(h_e)$$
$$=\{self\}\,.\,head_{rel}(h_s)\,.\,next_{rel}(h_s)^{*}\,.\,data_{rel}(h_s) \cup \{d\}$$

$$h_e = store(store(store(create(h_s, e), self, \mathtt{head}, e), e, \mathtt{next}, h_s[self.\mathtt{head}]), e, \mathtt{data}, d)$$

- Relational heap resolution

$$head_{rel}(store(h_4, e, \mathtt{data}, d)) \rightsquigarrow$$

Heap Resolution Rules – *Example*

- Postcondition of `prepend(d)` (in Alloy)

$$self\,.\,head'\,.\,next'^{*}\,.\,data' = self\,.\,head\,.\,next^{*}\,.\,data \cup d$$

- In relational JavaDL (KeY's logic)

$$\{self\}\,.\,head_{rel}(h_e)\,.\,next_{rel}(h_e)^{*}\,.\,data_{rel}(h_e)$$
$$=\{self\}\,.\,head_{rel}(h_s)\,.\,next_{rel}(h_s)^{*}\,.\,data_{rel}(h_s) \cup \{d\}$$

$h_e = store(store(store(create(h_s, e), self, \texttt{head}, e), e, \texttt{next}, h_s[self.\texttt{head}]), e, \texttt{data}, d)$

- Relational heap resolution

$head_{rel}(store(h_4, e, \texttt{data}, d)) \rightsquigarrow$

Heap Resolution Rules – *Example*

- Postcondition of `prepend(d)` (in Alloy)

$$self \,.\, head' \,.\, next'^{*} \,.\, data' = self \,.\, head \,.\, next^{*} \,.\, data \cup d$$

- In relational JavaDL (KeY's logic)

$$\{self\} \,.\, head_{rel}(h_e) \,.\, next_{rel}(h_e)^{*} \,.\, data_{rel}(h_e)$$
$$= \{self\} \,.\, head_{rel}(h_s) \,.\, next_{rel}(h_s)^{*} \,.\, data_{rel}(h_s) \cup \{d\}$$

$$h_e = store(store(store(create(h_s, e), self, \mathtt{head}, e), e, \mathtt{next}, h_s[self.\mathtt{head}]), e, \mathtt{data}, d)$$

- Relational heap resolution

$$head_{rel}(store(h_4, e, \mathtt{data}, d)) \rightsquigarrow head_{rel}(store(h_3, e, \mathtt{next}, h_s[self.\mathtt{head}]))$$

Heap Resolution Rules – *Example*

- Postcondition of `prepend(d)` (in Alloy)

$$self \,.\, head' \,.\, next'^{*} \,.\, data' = self \,.\, head \,.\, next^{*} \,.\, data \cup d$$

- In relational JavaDL (KeY's logic)

$$\{self\} \,.\, head_{rel}(h_e) \,.\, next_{rel}(h_e)^{*} \,.\, data_{rel}(h_e)$$
$$= \{self\} \,.\, head_{rel}(h_s) \,.\, next_{rel}(h_s)^{*} \,.\, data_{rel}(h_s) \cup \{d\}$$

$$h_e = store(store(store(create(h_s, e), self, \mathtt{head}, e), e, \mathtt{next}, h_s[self.\mathtt{head}]), e, \mathtt{data}, d)$$

- Relational heap resolution

$$head_{rel}(store(h_3, e, \mathtt{next}, h_s[self.\mathtt{head}])) \rightsquigarrow$$

Heap Resolution Rules – *Example*

- Postcondition of `prepend(d)` (in Alloy)

$$self\,.\,head'\,.\,next'^{*}\,.\,data' = self\,.\,head\,.\,next^{*}\,.\,data \cup d$$

- In relational JavaDL (KeY's logic)

$$\begin{aligned}&\{self\}\,.\,head_{rel}(h_e)\,.\,next_{rel}(h_e)^{*}\,.\,data_{rel}(h_e)\\=&\{self\}\,.\,head_{rel}(h_s)\,.\,next_{rel}(h_s)^{*}\,.\,data_{rel}(h_s) \cup \{d\}\end{aligned}$$

$$h_e = store(store(store(create(h_s, e), self, \mathtt{head}, e), e, \mathtt{next}, h_s[self.\mathtt{head}]), e, \mathtt{data}, d)$$

- Relational heap resolution

$$head_{rel}(store(h_3, e, \mathtt{next}, h_s[self.\mathtt{head}])) \rightsquigarrow head_{rel}(store(h_2, self, \mathtt{head}, e))$$

Heap Resolution Rules – *Example*

- Postcondition of `prepend(d)` (in Alloy)

$$self \,.\, head' \,.\, next'^{*} \,.\, data' = self \,.\, head \,.\, next^{*} \,.\, data \cup d$$

- In relational JavaDL (KeY's logic)

$$\{self\} \,.\, head_{rel}(h_e) \,.\, next_{rel}(h_e)^{*} \,.\, data_{rel}(h_e)$$
$$=\{self\} \,.\, head_{rel}(h_s) \,.\, next_{rel}(h_s)^{*} \,.\, data_{rel}(h_s) \cup \{d\}$$

$$h_e = store(store(store(create(h_s, e), self, \mathtt{head}, e), e, \mathtt{next}, h_s[self.\mathtt{head}]), e, \mathtt{data}, d)$$

- Relational heap resolution

$$head_{rel}(store(h_2, self, \mathtt{head}, e)) \rightsquigarrow$$

Heap Resolution Rules – *Example*

- Postcondition of `prepend(d)` (in Alloy)

$$self \,.\, head' \,.\, next'^{*} \,.\, data' = self \,.\, head \,.\, next^{*} \,.\, data \cup d$$

- In relational JavaDL (KeY's logic)

$$\{self\} \,.\, head_{rel}(h_e) \,.\, next_{rel}(h_e)^{*} \,.\, data_{rel}(h_e)$$
$$= \{self\} \,.\, head_{rel}(h_s) \,.\, next_{rel}(h_s)^{*} \,.\, data_{rel}(h_s) \cup \{d\}$$

$$h_e = store(store(store(create(h_s, e), self, \mathtt{head}, e), e, \mathtt{next}, h_s[self.\mathtt{head}]), e, \mathtt{data}, d)$$

- Relational heap resolution

$$head_{rel}(store(h_2, self, \mathtt{head}, e)) \rightsquigarrow head_{rel}(create(h_s, e)) \oplus \{self\} \times \{e\}$$

Heap Resolution Rules – *Example*

- Postcondition of `prepend(d)` (in Alloy)

$$self \,.\, head' \,.\, next'^{*} \,.\, data' = self \,.\, head \,.\, next^{*} \,.\, data \cup d$$

- In relational JavaDL (KeY's logic)

$$\begin{aligned}&\{self\} \,.\, head_{rel}(h_e) \,.\, next_{rel}(h_e)^{*} \,.\, data_{rel}(h_e)\\ =&\{self\} \,.\, head_{rel}(h_s) \,.\, next_{rel}(h_s)^{*} \,.\, data_{rel}(h_s) \cup \{d\}\end{aligned}$$

$$h_e = store(store(store(create(h_s, e), self, \mathtt{head}, e), e, \mathtt{next}, h_s[self.\mathtt{head}]), e, \mathtt{data}, d)$$

- Relational heap resolution

$$\begin{aligned}head_{rel}(h_e) &\overset{\cdots}{\rightsquigarrow} head_{rel}(h_s) \oplus \{self\} \times \{e\}\\ next_{rel}(h_e) &\overset{\cdots}{\rightsquigarrow} next_{rel}(h_s) \oplus \{e\} \times \{self\} \,.\, head_{rel}(h_s)\\ data_{rel}(h_e) &\overset{\cdots}{\rightsquigarrow} data_{rel}(h_s) \oplus \{e\} \times \{d\}\end{aligned}$$

Heap Resolution Rules – *Example*

- Postcondition of `prepend(d)` (in Alloy)

$$self \,.\, head' \,.\, next'^{*} \,.\, data' = self \,.\, head \,.\, next^{*} \,.\, data \cup d$$

- In relational JavaDL (KeY's logic)

$$\begin{aligned}&\{self\} \,.\, head_{rel}(h_e) \,.\, next_{rel}(h_e)^{*} \,.\, data_{rel}(h_e)\\ =&\{self\} \,.\, head_{rel}(h_s) \,.\, next_{rel}(h_s)^{*} \,.\, data_{rel}(h_s) \cup \{d\}\end{aligned}$$

$$h_e = store(store(store(create(h_s, e), self, \mathtt{head}, e), e, \mathtt{next}, h_s[self.\mathtt{head}]), e, \mathtt{data}, d)$$

- Postcondition after Heap Resolution

$$\begin{aligned}&\{self\} \,.\, (head_{rel}(h_s) \oplus \{self\} \times \{e\})\\ &.\, (next_{rel}(h_s) \oplus \{e\} \times \{self\} \,.\, head_{rel}(h_s))^{*} \,.\, (data_{rel}(h_s) \oplus \{e\} \times \{d\})\\ =&\{self\} \,.\, head_{rel}(h_s) \,.\, next_{rel}(h_s)^{*} \,.\, data_{rel}(h_s) \cup \{d\}\end{aligned}$$

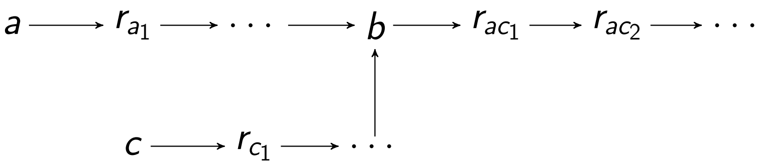
Override Resolution Rules: Example 1

$\mathbf{R_1}$**:** $\{a\} . (R \oplus \{a\} \times \{b\}) \rightsquigarrow \{b\}$

Override Resolution Rules: Example 1

$\mathbf{R}_1$**:** $\{a\} . (R \oplus \{a\} \times \{b\}) \rightsquigarrow \{b\}$

Override Resolution Rules: Example 2

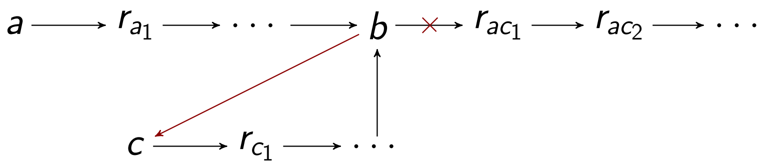


$$\mathbf{R_3:}\ \{\mathbf{a}\} \,.\, (\mathbf{R} \oplus \{\mathbf{b}\} \times \{\mathbf{c}\})^+ \rightsquigarrow$$

$$\begin{array}{ll} \text{if } b \in \{c\}.R^+ \vee b = c & \text{then } (\{\mathbf{a}\} \,.\, \mathbf{R}^+ \cup \{\mathbf{c}\} \cup \{\mathbf{c}\} \,.\, \mathbf{R}^+) \setminus \{\mathbf{b}\} \,.\, \mathbf{R}^+ \\ & \text{else } (\{\mathbf{a}\} \,.\, \mathbf{R}^+ \setminus \{\mathbf{b}\} \,.\, \mathbf{R}^+) \cup \{\mathbf{c}\} \cup \{\mathbf{c}\} \,.\, \mathbf{R}^+ \end{array}$$

assuming $b \in \{a\} \,.\, R^+$, $parFun(R)$ and $acyc(R)$

Override Resolution Rules: Example 2



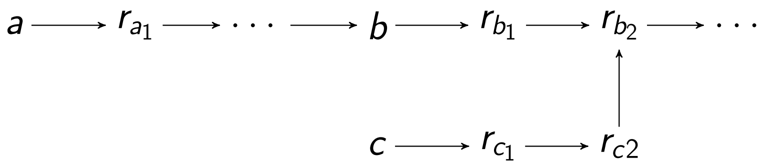
$\mathbf{R_3}$: $\{\mathbf{a}\} \,.\, (\mathbf{R} \oplus \{\mathbf{b}\} \times \{\mathbf{c}\})^+ \leadsto$

if $b \in \{c\}.R^+ \vee b = c$ then $(\{\mathbf{a}\} \,.\, \mathbf{R}^+ \cup \{\mathbf{c}\} \cup \{\mathbf{c}\} \,.\, \mathbf{R}^+) \setminus \{\mathbf{b}\} \,.\, \mathbf{R}^+$

else $(\{\mathbf{a}\} \,.\, \mathbf{R}^+ \setminus \{\mathbf{b}\} \,.\, \mathbf{R}^+) \cup \{\mathbf{c}\} \cup \{\mathbf{c}\} \,.\, \mathbf{R}^+$

assuming $b \in \{a\} \,.\, R^+$, $parFun(R)$ and $acyc(R)$

Override Resolution Rules: Example 2

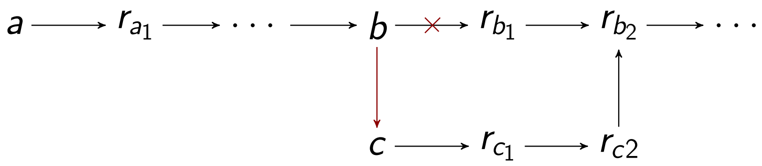


$\mathbf{R}_3$**:** $\{\mathbf{a}\}\,.\,(\mathbf{R} \oplus \{\mathbf{b}\} \times \{\mathbf{c}\})^+ \rightsquigarrow$

$$\text{if } b \in \{c\}.R^+ \vee b = c \quad \text{then } (\{\mathbf{a}\}\,.\,\mathbf{R}^+ \cup \{\mathbf{c}\} \cup \{\mathbf{c}\}\,.\,\mathbf{R}^+) \setminus \{\mathbf{b}\}\,.\,\mathbf{R}^+$$
$$\text{else } (\{\mathbf{a}\}\,.\,\mathbf{R}^+ \setminus \{\mathbf{b}\}\,.\,\mathbf{R}^+) \cup \{\mathbf{c}\} \cup \{\mathbf{c}\}\,.\,\mathbf{R}^+$$

assuming $b \in \{a\}\,.\,R^+$, *parFun*(R) and *acyc*(R)

Override Resolution Rules: Example 2



$$\mathbf{R_3}\mathbf{:}\ \{\mathbf{a}\} \,.\, (\mathbf{R} \oplus \{\mathbf{b}\} \times \{\mathbf{c}\})^{+} \rightsquigarrow$$

$$\text{if } b \in \{c\}.R^{+} \lor b = c \quad \text{then } (\{\mathbf{a}\} \,.\, \mathbf{R}^{+} \cup \{\mathbf{c}\} \cup \{\mathbf{c}\} \,.\, \mathbf{R}^{+}) \setminus \{\mathbf{b}\} \,.\, \mathbf{R}^{+}$$

$$\text{else } (\{\mathbf{a}\} \,.\, \mathbf{R}^{+} \setminus \{\mathbf{b}\} \,.\, \mathbf{R}^{+}) \cup \{\mathbf{c}\} \cup \{\mathbf{c}\} \,.\, \mathbf{R}^{+}$$

assuming $b \in \{a\} \,.\, R^{+}$, $parFun(R)$ and $acyc(R)$

Proof of prepend using Override Resolution Rules

Postcondition of 'prepend(d)' after heap resolution

Proof of prepend using Override Resolution Rules

Postcondition of 'prepend(d)' after heap resolution

Proof of `prepend` using Override Resolution Rules

Postcondition of 'prepend(d)' after heap resolution

$$\{self\} \,.\, (head_{rel}(h_s) \oplus \{self\} \times \{e\}) \,.\, (next_{rel}(h_s) \oplus \{e\} \times \{self\} \,.\, head_{rel}(h_s))^* \,.\, (data_{rel}(h_s) \oplus \{e\} \times \{d\})$$

$$= \{self\} \,.\, head_{rel}(h_s) \,.\, next_{rel}(h_s)^* \,.\, data_{rel}(h_s) \cup \{d\}$$

Proof of prepend using Override Resolution Rules

Postcondition of 'prepend(d)' after heap resolution

$$
\begin{aligned}
&\{self\} . (head_{rel}(h_s) \oplus \{self\} \times \{e\}) \\
&. (next_{rel}(h_s) \oplus \{e\} \times \{self\} . head_{rel}(h_s))^* . (data_{rel}(h_s) \oplus \{e\} \times \{d\}) \\
= &\{self\} . head_{rel}(h_s) . next_{rel}(h_s)^* . data_{rel}(h_s) \cup \{d\}
\end{aligned}
$$

Proof of prepend using Override Resolution Rules

Postcondition of 'prepend(d)' after heap resolution

$$\{self\} \,.\, (head_{rel}(h_s) \oplus \{self\} \times \{e\})$$
$$.\, (next_{rel}(h_s) \oplus \{e\} \times \{self\} \,.\, head_{rel}(h_s))^* \,.\, (data_{rel}(h_s) \oplus \{e\} \times \{d\})$$
$$= \{self\} \,.\, head_{rel}(h_s) \,.\, next_{rel}(h_s)^* \,.\, data_{rel}(h_s) \cup \{d\}$$

Proof of prepend using Override Resolution Rules

Postcondition of 'prepend(d)' after heap resolution

$$\{self\} \,.\, (head_{rel}(h_s) \oplus \{self\} \times \{e\})$$
$$.\, (next_{rel}(h_s) \oplus \{e\} \times \{self\} \,.\, head_{rel}(h_s))^* \,.\, (data_{rel}(h_s) \oplus \{e\} \times \{d\})$$
$$= \{self\} \,.\, head_{rel}(h_s) \,.\, next_{rel}(h_s)^* \,.\, data_{rel}(h_s) \cup \{d\}$$

Proof of `prepend` using Override Resolution Rules

Postcondition of 'prepend(d)' after heap resolution

$$\{self\} \,.\, (head_{rel}(h_s) \oplus \{self\} \times \{e\})$$
$$.\, (next_{rel}(h_s) \oplus \{e\} \times \{self\} \,.\, head_{rel}(h_s))^* \,.\, (data_{rel}(h_s) \oplus \{e\} \times \{d\})$$
$$= \{self\} \,.\, head_{rel}(h_s) \,.\, next_{rel}(h_s)^* \,.\, data_{rel}(h_s) \cup \{d\}$$

Proof of prepend using Override Resolution Rules

Postcondition of 'prepend(d)' after heap resolution

$$\{e\} . (next_{rel}(h_s) \oplus \{e\} \times \{self\} . head_{rel}(h_s))^* . (data_{rel}(h_s) \oplus \{e\} \times \{d\})$$
$$=\{self\} . head_{rel}(h_s) . next_{rel}(h_s)^* . data_{rel}(h_s) \cup \{d\}$$

Proof of `prepend` using Override Resolution Rules

Postcondition of 'prepend(d)' after heap resolution

$$\{e\} \mathbin{.} (next_{rel}(h_s) \oplus \{e\} \times \{self\} \mathbin{.} head_{rel}(h_s))^* \mathbin{.} (data_{rel}(h_s) \oplus \{e\} \times \{d\})$$
$$= \{self\} \mathbin{.} head_{rel}(h_s) \mathbin{.} next_{rel}(h_s)^* \mathbin{.} data_{rel}(h_s) \cup \{d\}$$

Proof of prepend using Override Resolution Rules

Postcondition of 'prepend(d)' after heap resolution

$$(\{e\} \cup \{e\} . (next_{rel}(h_s) \oplus \{e\} \times \{self\} . head_{rel}(h_s))^{+}) . (data_{rel}(h_s) \oplus \{e\} \times \{d\})$$
$$= (\{self\} . head_{rel}(h_s) \cup \{self\} . head_{rel}(h_s) . next_{rel}(h_s)^{+}) . data_{rel}(h_s) \cup \{d\}$$

Proof of prepend using Override Resolution Rules

Postcondition of 'prepend(d)' after heap resolution

$$(\{e\} \cup \{e\} . (next_{rel}(h_s) \oplus \{e\} \times \{self\} . head_{rel}(h_s))^{+}) . (data_{rel}(h_s) \oplus \{e\} \times \{d\})$$
$$= (\{self\} . head_{rel}(h_s) \cup \{self\} . head_{rel}(h_s) . next_{rel}(h_s)^{+}) . data_{rel}(h_s) \cup \{d\}$$

Proof of prepend using Override Resolution Rules

Postcondition of 'prepend(d)' after heap resolution

$$
\begin{aligned}
&\{e\} \,.\, (data_{rel}(h_s) \oplus \{e\} \times \{d\}) \cup \\
&\{e\} \,.\, (next_{rel}(h_s) \oplus \{e\} \times \{self\} \,.\, head_{rel}(h_s))^{+} \,.\, (data_{rel}(h_s) \oplus \{e\} \times \{d\}) \\
=&\{self\} \,.\, head_{rel}(h_s) \,.\, data_{rel}(h_s) \cup \\
&\{self\} \,.\, head_{rel}(h_s) \,.\, next_{rel}(h_s)^{+} \,.\, data_{rel}(h_s) \cup \\
&\{d\}
\end{aligned}
$$

Proof of prepend using Override Resolution Rules

Postcondition of 'prepend(d)' after heap resolution

$$\{e\} . (data_{rel}(h_s) \oplus \{e\} \times \{d\}) \cup$$
$$\{e\} . (next_{rel}(h_s) \oplus \{e\} \times \{self\} . head_{rel}(h_s))^+ . (data_{rel}(h_s) \oplus \{e\} \times \{d\})$$
$$= \{self\} . head_{rel}(h_s) . data_{rel}(h_s) \cup$$
$$\{self\} . head_{rel}(h_s) . next_{rel}(h_s)^+ . data_{rel}(h_s) \cup$$
$$\{d\}$$

Proof of prepend using Override Resolution Rules

Postcondition of 'prepend(d)' after heap resolution

$$
\begin{aligned}
& \{d\} \cup \\
& \{e\} . (next_{rel}(h_s) \oplus \{e\} \times \{self\} . head_{rel}(h_s))^{+} . (data_{rel}(h_s) \oplus \{e\} \times \{d\}) \\
= & \{self\} . head_{rel}(h_s) . data_{rel}(h_s) \cup \\
& \{self\} . head_{rel}(h_s) . next_{rel}(h_s)^{+} . data_{rel}(h_s) \cup \\
& \{d\}
\end{aligned}
$$

Proof of `prepend` using Override Resolution Rules

Postcondition of 'prepend(d)' after heap resolution

$$
\begin{aligned}
&\{d\} \cup \\
&\{e\} \,.\, (next_{rel}(h_s) \oplus \{e\} \times \{self\} \,.\, head_{rel}(h_s))^{+} \,.\, (data_{rel}(h_s) \oplus \{e\} \times \{d\}) \\
=&\{self\} \,.\, head_{rel}(h_s) \,.\, data_{rel}(h_s) \cup \\
&\{self\} \,.\, head_{rel}(h_s) \,.\, next_{rel}(h_s)^{+} \,.\, data_{rel}(h_s) \cup \\
&\{d\}
\end{aligned}
$$

Proof of `prepend` using Override Resolution Rules

Postcondition of 'prepend(d)' after heap resolution

$$
\begin{aligned}
& \{d\} \cup \\
& (\{self\} \,.\, head_{rel}(h_s) \cup \{self\} \,.\, head_{rel}(h_s) \,.\, next_{rel}(h_s)^{+}) \,.\, (data_{rel}(h_s) \oplus \{e\} \times \{d\}) \\
= & \{self\} \,.\, head_{rel}(h_s) \,.\, data_{rel}(h_s) \cup \\
& \{self\} \,.\, head_{rel}(h_s) \,.\, next_{rel}(h_s)^{+} \,.\, data_{rel}(h_s) \cup \\
& \{d\}
\end{aligned}
$$

Proof of `prepend` using Override Resolution Rules

Postcondition of 'prepend(d)' after heap resolution

$$
\begin{aligned}
&\{d\} \cup \\
&(\{self\} \,.\, head_{rel}(h_s) \cup \{self\} \,.\, head_{rel}(h_s) \,.\, next_{rel}(h_s)^{+}) \,.\, (data_{rel}(h_s) \oplus \{e\} \times \{d\}) \\
=&\{self\} \,.\, head_{rel}(h_s) \,.\, data_{rel}(h_s) \cup \\
&\{self\} \,.\, head_{rel}(h_s) \,.\, next_{rel}(h_s)^{+} \,.\, data_{rel}(h_s) \cup \\
&\{d\}
\end{aligned}
$$

Proof of prepend using Override Resolution Rules

Postcondition of 'prepend(d)' after heap resolution

$$
\begin{aligned}
&\{d\} \cup \\
&\{self\} \,.\, head_{rel}(h_s) \,.\, (data_{rel}(h_s) \oplus \{e\} \times \{d\}) \cup \\
&\{self\} \,.\, head_{rel}(h_s) \,.\, next_{rel}(h_s)^{+} \,.\, (data_{rel}(h_s) \oplus \{e\} \times \{d\}) \\
=&\{self\} \,.\, head_{rel}(h_s) \,.\, data_{rel}(h_s) \cup \\
&\{self\} \,.\, head_{rel}(h_s) \,.\, next_{rel}(h_s)^{+} \,.\, data_{rel}(h_s) \cup \\
&\{d\}
\end{aligned}
$$

Proof of `prepend` using Override Resolution Rules

Postcondition of 'prepend(d)' after heap resolution

$$
\begin{aligned}
& \{d\} \cup \\
& \{self\} \,.\, head_{rel}(h_s) \,.\, (data_{rel}(h_s) \oplus \{e\} \times \{d\}) \cup \\
& \{self\} \,.\, head_{rel}(h_s) \,.\, next_{rel}(h_s)^{+} \,.\, (data_{rel}(h_s) \oplus \{e\} \times \{d\}) \\
= & \{self\} \,.\, head_{rel}(h_s) \,.\, data_{rel}(h_s) \cup \\
& \{self\} \,.\, head_{rel}(h_s) \,.\, next_{rel}(h_s)^{+} \,.\, data_{rel}(h_s) \cup \\
& \{d\}
\end{aligned}
$$

Proof of prepend using Override Resolution Rules

Postcondition of 'prepend(d)' after heap resolution

$$
\begin{aligned}
&\{d\} \cup \\
&\{self\} \,.\, head_{rel}(h_s) \,.\, data_{rel}(h_s) \cup \\
&\{self\} \,.\, head_{rel}(h_s) \,.\, next_{rel}(h_s)^+ \,.\, (data_{rel}(h_s) \oplus \{e\} \times \{d\}) \\
=&\{self\} \,.\, head_{rel}(h_s) \,.\, data_{rel}(h_s) \cup \\
&\{self\} \,.\, head_{rel}(h_s) \,.\, next_{rel}(h_s)^+ \,.\, data_{rel}(h_s) \cup \\
&\{d\}
\end{aligned}
$$

Proof of prepend using Override Resolution Rules

Postcondition of 'prepend(d)' after heap resolution

$$
\begin{aligned}
& \{d\} \cup \\
& \{self\} \mathbin{.} head_{rel}(h_s) \mathbin{.} data_{rel}(h_s) \cup \\
& \{self\} \mathbin{.} head_{rel}(h_s) \mathbin{.} next_{rel}(h_s)^{+} \mathbin{.} (data_{rel}(h_s) \oplus \{e\} \times \{d\}) \\
= & \{self\} \mathbin{.} head_{rel}(h_s) \mathbin{.} data_{rel}(h_s) \cup \\
& \{self\} \mathbin{.} head_{rel}(h_s) \mathbin{.} next_{rel}(h_s)^{+} \mathbin{.} data_{rel}(h_s) \cup \\
& \{d\}
\end{aligned}
$$

Proof of prepend using Override Resolution Rules

Postcondition of 'prepend(d)' after heap resolution

$$
\begin{aligned}
&\{d\} \cup \\
&\{self\} . head_{rel}(h_s) . data_{rel}(h_s) \cup \\
&\{self\} . head_{rel}(h_s) . next_{rel}(h_s)^+ . data_{rel}(h_s) \\
= &\{self\} . head_{rel}(h_s) . data_{rel}(h_s) \cup \\
&\{self\} . head_{rel}(h_s) . next_{rel}(h_s)^+ . data_{rel}(h_s) \cup \\
&\{d\}
\end{aligned}
$$

Proof of `prepend` using Override Resolution Rules

Postcondition of 'prepend(d)' after heap resolution

$$
\begin{aligned}
&\{d\} \cup \\
&\{self\} \mathbin{.} head_{rel}(h_s) \mathbin{.} data_{rel}(h_s) \cup \\
&\{self\} \mathbin{.} head_{rel}(h_s) \mathbin{.} next_{rel}(h_s)^{+} \mathbin{.} data_{rel}(h_s) \\
=&\{self\} \mathbin{.} head_{rel}(h_s) \mathbin{.} data_{rel}(h_s) \cup \\
&\{self\} \mathbin{.} head_{rel}(h_s) \mathbin{.} next_{rel}(h_s)^{+} \mathbin{.} data_{rel}(h_s) \cup \\
&\{d\}
\end{aligned}
$$

Proof of prepend using Override Resolution Rules

Postcondition of 'prepend(d)' after heap resolution

$$
\begin{aligned}
&\{self\}\,.\,head_{rel}(h_s)\,.\,data_{rel}(h_s) \cup \\
&\{self\}\,.\,head_{rel}(h_s)\,.\,next_{rel}(h_s)^{+}\,.\,data_{rel}(h_s) \cup \\
&\{d\} \\
=&\{self\}\,.\,head_{rel}(h_s)\,.\,data_{rel}(h_s) \cup \\
&\{self\}\,.\,head_{rel}(h_s)\,.\,next_{rel}(h_s)^{+}\,.\,data_{rel}(h_s) \cup \\
&\{d\}
\end{aligned}
$$

More Override Resolution Rules

Further rules to reason about the assumptions and conditions of the main rules

- Partial funtionality preservation, e.g.
  $\mathbf{R}_8$: $\vdash \mathit{parFun}(\mathbf{R_1}) \wedge \mathit{parFun}(\mathbf{R_2}) \rightarrow \mathit{parFun}(\mathbf{R_1} \oplus \mathbf{R_2})$
- Acyclicity preservation, e.g.
  $\mathbf{R}_9$: $\vdash \mathit{acyc}(\mathbf{R}) \wedge \mathbf{R} \,.\, \{\mathbf{a}\} = \emptyset \wedge \mathbf{a} \neq \mathbf{b} \rightarrow \mathit{acyc}(\mathbf{R} \oplus \{\mathbf{a}\} \times \{\mathbf{b}\})$
- Reachability lemmas, e.g.
  $\mathbf{R}_{12}$: $S_2 \in S_1 \,.\, R^+ \rightsquigarrow \mathbf{false}$ assuming $S_1 \,.\, R = \emptyset$
- See TR for the general fragment supported by the override simplification rules

Evaluation

- `List.prepend` fully automatic, 1546 RuleApps, 5.4 sec
- `List.append`, 28 interactive, 2850 RuleApps
- `Graph.remove`, 1201 interactive, 6973 RuleApps

Potential to optimize proof strategy

Conclusion

- Deductive verification of Java programs, concise Alloy specs
- Problem we addressed: How does a data structure change as a whole, when modifying a Java field
- Concise proofs of linked data structure without induction
- Coupling Axioms, 2 sets of simplification rules
- Evaluation: promissing approach; early development stage