Automatically Finding Non-confluent Examples in Abstract Rewriting

Hans Zantema

Technische Universiteit Eindhoven and Radboud Universiteit Nijmegen

IWC, Nagoya, May 29, 2012

Our goal:

Our goal:

Given a number of abstract rewrite properties and a number n, find a model of n elements satisfying these properties

Our goal:

Given a number of abstract rewrite properties and a number n, find a model of n elements satisfying these properties

Example:

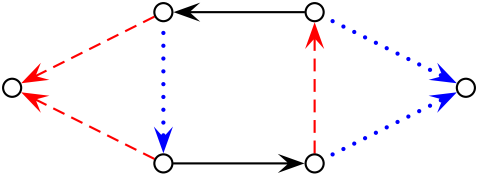
Can we find three rewrite relations such that the union of any two of them is both terminating and confluent, but the union of all three is not confluent?

Our goal:

Given a number of abstract rewrite properties and a number n, find a model of n elements satisfying these properties

Example:
Can we find three rewrite relations such that the union of any two of them is both terminating and confluent, but the union of all three is not confluent?

Solution for $n = 6$:



in which the three relations are indicated by solid black, dashed red and dotted blue arrows, respectively

Main approach

Main approach

- Fix the number n

Main approach

- Fix the number n
- For every rewrite relation introduce n^2 boolean variables R_{ij} indicating whether (i, j) is in the relation or not

Main approach

- Fix the number n
- For every rewrite relation introduce n^2 boolean variables R_{ij} indicating whether (i, j) is in the relation or not
- Express all requirements by propositional formulas in the boolean variables

Main approach

- Fix the number n
- For every rewrite relation introduce n^2 boolean variables R_{ij} indicating whether (i, j) is in the relation or not
- Express all requirements by propositional formulas in the boolean variables
- Apply a SAT solver on the result

Main approach

- Fix the number n
- For every rewrite relation introduce n^2 boolean variables R_{ij} indicating whether (i, j) is in the relation or not
- Express all requirements by propositional formulas in the boolean variables
- Apply a SAT solver on the result
- In case of satisfiability: extract the solution from the resulting satisfying assignment

Main questions

Main questions

How to express the standard rewrite properties like

Main questions

How to express the standard rewrite properties like

- Termination

Main questions

How to express the standard rewrite properties like

- Termination
- Confluence

Main questions

How to express the standard rewrite properties like

- Termination
- Confluence
- Normal forms

Main questions

How to express the standard rewrite properties like

- Termination
- Confluence
- Normal forms
- Commutation properties like

$$R^{-1} \cdot S \subseteq (S \cup R^*) \cdot (R^{-1})^*$$

Main questions

How to express the standard rewrite properties like

- Termination
- Confluence
- Normal forms
- Commutation properties like

$$R^{-1} \cdot S \subseteq (S \cup R^*) \cdot (R^{-1})^*$$

For expressing most of these properties auxiliary relations will be needed

Termination

Theorem

A binary relation R on a finite set is terminating if and only a binary relation S on the same set exists that is transitive and irreflexive, and for which $R \subseteq S$

Termination

Theorem

A binary relation R on a finite set is terminating if and only a binary relation S on the same set exists that is transitive and irreflexive, and for which $R \subseteq S$

Proof:

Theorem

A binary relation R on a finite set is terminating if and only a binary relation S on the same set exists that is transitive and irreflexive, and for which $R \subseteq S$

Proof:

(only if)
Choose $S = R^+$, this is irreflexive since R is acyclic

Theorem

A binary relation R on a finite set is terminating if and only a binary relation S on the same set exists that is transitive and irreflexive, and for which $R \subseteq S$

Proof:

(only if)
Choose $S = R^+$, this is irreflexive since R is acyclic

(if)
If S is transitive and $R \subseteq S$, then $R^+ \subseteq S$

Theorem

A binary relation R on a finite set is terminating if and only a binary relation S on the same set exists that is transitive and irreflexive, and for which $R \subseteq S$

Proof:

(only if)
Choose $S = R^+$, this is irreflexive since R is acyclic

(if)
If S is transitive and $R \subseteq S$, then $R^+ \subseteq S$

Hence R^+ is irreflexive, hence R is acyclic = terminating

Theorem

A binary relation R on a finite set is terminating if and only a binary relation S on the same set exists that is transitive and irreflexive, and for which $R \subseteq S$

Termination

Theorem

A binary relation R on a finite set is terminating if and only a binary relation S on the same set exists that is transitive and irreflexive, and for which $R \subseteq S$

Using this theorem the requirement of termination of a relation R can be expressed in a SAT formula:

Termination

Theorem

A binary relation R on a finite set is terminating if and only a binary relation S on the same set exists that is transitive and irreflexive, and for which $R \subseteq S$

Using this theorem the requirement of termination of a relation R can be expressed in a SAT formula:

Introduce an auxiliary relation S by n^2 extra variables S_{ij} and express

Termination

Theorem

A binary relation R on a finite set is terminating if and only a binary relation S on the same set exists that is transitive and irreflexive, and for which $R \subseteq S$

Using this theorem the requirement of termination of a relation R can be expressed in a SAT formula:

Introduce an auxiliary relation S by n^2 extra variables S_{ij} and express

- $R \subseteq S$ by $\bigwedge_{i,j}(R_{ij} \rightarrow S_{ij})$

Theorem

A binary relation R on a finite set is terminating if and only a binary relation S on the same set exists that is transitive and irreflexive, and for which $R \subseteq S$

Using this theorem the requirement of termination of a relation R can be expressed in a SAT formula:

Introduce an auxiliary relation S by n^2 extra variables S_{ij} and express

- $R \subseteq S$ by $\bigwedge_{i,j}(R_{ij} \rightarrow S_{ij})$
- S is irreflexive by $\bigwedge_i \neg S_{ii}$

Termination

Theorem

A binary relation R on a finite set is terminating if and only a binary relation S on the same set exists that is transitive and irreflexive, and for which $R \subseteq S$

Using this theorem the requirement of termination of a relation R can be expressed in a SAT formula:

Introduce an auxiliary relation S by n^2 extra variables S_{ij} and express

- $R \subseteq S$ by $\bigwedge_{i,j}(R_{ij} \rightarrow S_{ij})$
- S is irreflexive by $\bigwedge_i \neg S_{ii}$
- S is transitive by $\bigwedge_{i,j,k}((S_{ij} \wedge S_{jk}) \rightarrow S_{ik})$

Composition

Composition

$T \;=\; R \cdot S$ is expressed by

$$\bigwedge_{i,j}(T_{ij} \leftrightarrow \bigvee_{k}(R_{ik} \wedge S_{kj}))$$

Composition

$T = R \cdot S$ is expressed by

$$\bigwedge_{i,j} (T_{ij} \leftrightarrow \bigvee_{k} (R_{ik} \wedge S_{kj}))$$

In particular, we can express $T = R \cdot R = R^2$

Composition

$T = R \cdot S$ is expressed by

$$\bigwedge_{i,j}(T_{ij} \leftrightarrow \bigvee_k (R_{ik} \wedge S_{kj}))$$

In particular, we can express $T = R \cdot R = R^2$

Similarly, we express

$$peak(R,S) = R^{-1} \cdot S$$

and

$$valley(R,S) = R \cdot S^{-1}$$

Transitive reflexive closure

Theorem

Let R be a relation on a set of n elements and let $k = \lceil \log_2 n \rceil$

Transitive reflexive closure

Theorem

Let R be a relation on a set of n elements and let $k = \lceil \log_2 n \rceil$

Let R_i be relations for $i = 1, 2, \ldots, k$, satisfying

$$R_1 = I \cup R \cup R^2, \quad \textit{and} \quad R_{i+1} = R_i \cup R_i^2 \quad \textit{for } i = 1, \ldots, k-1$$

Then $R_k = R^*$

Transitive reflexive closure

Theorem

Let R be a relation on a set of n elements and let $k = \lceil \log_2 n \rceil$

Let R_i be relations for $i = 1, 2, \ldots, k$, satisfying

$$R_1 = I \cup R \cup R^2, \quad \textit{and} \quad R_{i+1} = R_i \cup R_i^2 \quad \textit{for } i = 1, \ldots, k-1$$

Then $R_k = R^*$

So for expressing the relation R^* for a given binary relation R introduce auxiliary relations $R_1, R_2, \ldots, R_k$ and create formulas expressing $R_1 = I \cup R \cup R^2$ for I being the identity, and $R_{i+1} = R_i \cup R_i^2$ for $i = 1, \ldots, k-1$

Transitive reflexive closure

Theorem

Let R be a relation on a set of n elements and let $k = \lceil \log_2 n \rceil$

Let R_i be relations for $i = 1, 2, \ldots, k$, satisfying

$$R_1 = I \cup R \cup R^2, \quad \textit{and} \quad R_{i+1} = R_i \cup R_i^2 \quad \textit{for } i = 1, \ldots, k-1$$

Then $R_k = R^*$

So for expressing the relation R^* for a given binary relation R introduce auxiliary relations $R_1, R_2, \ldots, R_k$ and create formulas expressing $R_1 = I \cup R \cup R^2$ for I being the identity, and $R_{i+1} = R_i \cup R_i^2$ for $i = 1, \ldots, k-1$

Then R_k describes the desired relation R^*

Confluence

Confluence

By specifying $S = R^*$ in this way, we express confluence by

$$\mathit{peak}(S, S) \ \subseteq \ \mathit{valley}(S, S)$$

and local confluence by

$$\mathit{peak}(R, R) \ \subseteq \ \mathit{valley}(S, S)$$

Confluence

By specifying $S = R^*$ in this way, we express confluence by

$$\mathit{peak}(S,S) \subseteq \mathit{valley}(S,S)$$

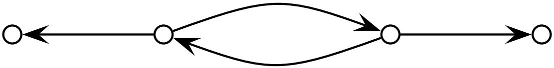
and local confluence by

$$\mathit{peak}(R,R) \subseteq \mathit{valley}(S,S)$$

Applying a SAT solver on the combination of

$$\neg(\mathit{peak}(S,S) \subseteq \mathit{valley}(S,S))$$

and local confluence for $n = 4$ yields the well-known example



Confluence

By specifying $S = R^*$ in this way, we express confluence by

$$peak(S, S) \subseteq valley(S, S)$$

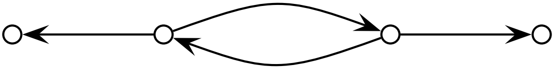
and local confluence by

$$peak(R, R) \subseteq valley(S, S)$$

Applying a SAT solver on the combination of

$$\neg(peak(S, S) \subseteq valley(S, S))$$

and local confluence for $n = 4$ yields the well-known example



Adding termination of R yields an unsatisfiable formula, as expected due to Newman's Lemma

Completeness

Completeness

Complete = confluent and terminating

Completeness

Complete = confluent and terminating

Completess can be expressed by the combination of confluence and termination using $\Omega(\log n)$ auxiliary relations

Completeness

Complete = confluent and terminating

Completess can be expressed by the combination of confluence and termination using $\Omega(\log n)$ auxiliary relations

It can be done much more efficient using only two auxiliary relations by

Theorem

A binary relation R on a finite set is complete if and only if two binary relations S and T exist such that

- $R \subseteq S$
- S *is transitive and irreflexive*
- $\bigwedge_i (T_{ii} \vee \bigvee_j R_{ij})$
- $\bigwedge_{i,j}((S_{ij} \wedge T_{jj}) \rightarrow T_{ij})$
- $\bigwedge_{i,j,k,j \neq k} \neg(T_{ij} \wedge T_{ik})$

The tool CARPA

The tool CARPA

We defined an input format in which rewriting properties can be specified directly:

The tool CARPA

We defined an input format in which rewriting properties can be specified directly:

- $\mathtt{sn}$, where $\mathtt{sn}(R)$ means that R is terminating,

The tool CARPA

We defined an input format in which rewriting properties can be specified directly:

- sn, where sn(R) means that R is terminating,
- cr, wcr, wn, un,

The tool CARPA

We defined an input format in which rewriting properties can be specified directly:

- `sn`, where $\texttt{sn}(R)$ means that R is terminating,
- `cr`, `wcr`, `wn`, `un`,
- `compl`, where $\texttt{compl}(R)$ means that R is complete,

The tool CARPA

We defined an input format in which rewriting properties can be specified directly:

- `sn`, where `sn`(R) means that R is terminating,
- `cr`, `wcr`, `wn`, `un`,
- `compl`, where `compl`(R) means that R is complete,
- `subs`, where `subs`(R, S) means that $R \subseteq S$,

The tool CARPA

We defined an input format in which rewriting properties can be specified directly:

- sn, where sn(R) means that R is terminating,
- cr, wcr, wn, un,
- compl, where compl(R) means that R is complete,
- subs, where subs(R, S) means that $R \subseteq S$,
- trans, where trans(R) means that R is transitive,

We defined an input format in which rewriting properties can be specified directly:

- `sn`, where $\mathtt{sn}(R)$ means that R is terminating,
- `cr`, `wcr`, `wn`, `un`,
- `compl`, where $\mathtt{compl}(R)$ means that R is complete,
- `subs`, where $\mathtt{subs}(R, S)$ means that $R \subseteq S$,
- `trans`, where $\mathtt{trans}(R)$ means that R is transitive,
- `nf`, where $\mathtt{nf}(x, R)$ means that x is a normal form with respect to R, and

The tool CARPA

We defined an input format in which rewriting properties can be specified directly:

- `sn`, where `sn`(R) means that R is terminating,
- `cr`, `wcr`, `wn`, `un`,
- `compl`, where `compl`(R) means that R is complete,
- `subs`, where `subs`(R, S) means that $R \subseteq S$,
- `trans`, where `trans`(R) means that R is transitive,
- `nf`, where `nf`(x, R) means that x is a normal form with respect to R, and
- `red`, where `red`(x, y, R) means that $(x, y) \in R$,

The tool CARPA

We defined an input format in which rewriting properties can be specified directly:

- `sn`, where `sn`(R) means that R is terminating,
- `cr`, `wcr`, `wn`, `un`,
- `compl`, where `compl`(R) means that R is complete,
- `subs`, where `subs`(R, S) means that $R \subseteq S$,
- `trans`, where `trans`(R) means that R is transitive,
- `nf`, where `nf`(x, R) means that x is a normal form with respect to R, and
- `red`, where `red`(x, y, R) means that $(x, y) \in R$,
- `union`, `comp`(osition), `peak`, `val`, transitive closure, $\cdots$

The tool CARPA

The tool CARPA

Our tool CARPA (Counter examples of Abstract Rewriting Produced Automatically)

The tool CARPA

Our tool CARPA (Counter examples of Abstract Rewriting Produced Automatically)

- reads the number *n* and a list of requirements in this format, like

```
cr(1)
x1=peak(1,2)
x2=trc(1)
x3=val(2,x2)
subs(x1,x3)
```

The tool CARPA

Our tool CARPA (Counter examples of Abstract Rewriting Produced Automatically)

- reads the number n and a list of requirements in this format, like

```
cr(1)
x1=peak(1,2)
x2=trc(1)
x3=val(2,x2)
subs(x1,x3)
```

- builds a formula for it,

The tool CARPA

Our tool CARPA (Counter examples of Abstract Rewriting Produced Automatically)

- reads the number *n* and a list of requirements in this format, like

```
cr(1)
x1=peak(1,2)
x2=trc(1)
x3=val(2,x2)
subs(x1,x3)
```

- builds a formula for it,
- calls a SAT solver, and

The tool CARPA

Our tool CARPA (Counter examples of Abstract Rewriting Produced Automatically)

- reads the number n and a list of requirements in this format, like

```
cr(1)
x1=peak(1,2)
x2=trc(1)
x3=val(2,x2)
subs(x1,x3)
```

- builds a formula for it,
- calls a SAT solver, and
- transforms the result back to the desired example, or reports that no solution exists

Example

Example

Let R and S be two complete binary relations satisfying

$$R^{-1} \cdot S \subseteq (S \cup R^*) \cdot (R^{-1})^*$$

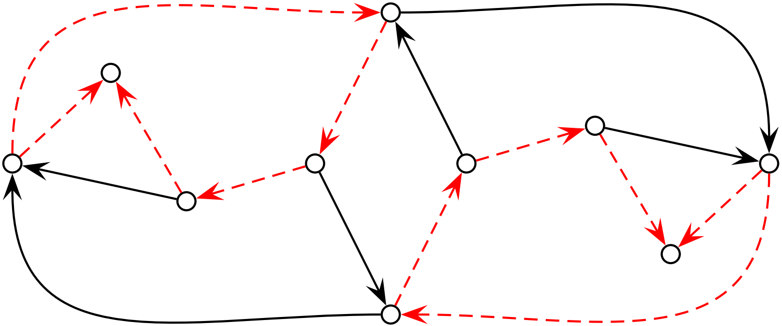
Can we conclude that the union is confluent?

Let R and S be two complete binary relations satisfying

$$R^{-1} \cdot S \ \subseteq \ (S \cup R^{*}) \cdot (R^{-1})^{*}$$

Can we conclude that the union is confluent?

No



R steps: solid black arrows; S steps: dashed red arrows

Conclusions

Conclusions

- We developed technigues to express abstract rewriting properties like termination, confluence, completeness, in propositional formulas

Conclusions

- We developed technigues to express abstract rewriting properties like termination, confluence, completeness, in propositional formulas
- We implemented this in a tool CARPA, that reads a list of rewriting properties and automatically finds an example, exploiting these techniques and SAT solving

Conclusions

- We developed technigues to express abstract rewriting properties like termination, confluence, completeness, in propositional formulas
- We implemented this in a tool CARPA, that reads a list of rewriting properties and automatically finds an example, exploiting these techniques and SAT solving
- As a SAT solver we experimented with `minisat` and `Yices`; the distributed version uses `Yices`

Conclusions

- We developed technigues to express abstract rewriting properties like termination, confluence, completeness, in propositional formulas
- We implemented this in a tool CARPA, that reads a list of rewriting properties and automatically finds an example, exploiting these techniques and SAT solving
- As a SAT solver we experimented with `minisat` and `Yices`; the distributed version uses `Yices`
- Termination and completeness were expressed by one or two auxiliary relations, for transitive closure and confluence we needed $\log n$ auxiliary relations

Conclusions

- We developed techniques to express abstract rewriting properties like termination, confluence, completeness, in propositional formulas
- We implemented this in a tool CARPA, that reads a list of rewriting properties and automatically finds an example, exploiting these techniques and SAT solving
- As a SAT solver we experimented with `minisat` and `Yices`; the distributed version uses `Yices`
- Termination and completeness were expressed by one or two auxiliary relations, for transitive closure and confluence we needed $\log n$ auxiliary relations
- More examples are welcome