

Symbolic Constraint Solving: Constraint Simplification vs. Automata

Ralf Treinen

Université Paris Diderot
Laboratoire Preuves, Programmes et Systèmes

January 19, 2008
Workshop in Honour of Hubert Comon

What is Symbolic Constraint Solving?

Does the following hold in the Herbrand universe over $\Sigma = \{f, g, a\}$?

$$\text{Herbrand}(f, g, a) \models \forall x \exists y, z (x = a \vee x = f(y, z) \vee x = g(y))$$

For which values of x is the following formula true :

$$\mathbb{N} \models x = 1 \vee \neg \exists y, z (x = y * (2z + 1))$$

What is Symbolic Constraint Solving?

- Deciding validity of first-order formulas in some *fixed* interpretation.
- Or: Simplifying first-order formulas into solved form, modulo some fixed interpretation.
- Many applications: programming, theorem proving, modelisation of infinite state systems, ...
- Often restricted to a special forms of formulas, because
 - Complete FO theory might be undecidable
 - Complexity might be too high
- But: deciding a full FO theory is the real challenge.

Constraint Simplification

Hubert Comon & Pierre Lescanne, 1988 : Constraint Simplification for Herbrand as extension of the unification rules:

$$(D) \quad \frac{f(t_1, \dots, t_n) = f(s_1, \dots, s_n)}{t_1 = s_1 \wedge \dots \wedge t_n = s_n}$$

$$(C) \quad \frac{f(t_1, \dots, t_n) = g(s_1, \dots, s_n)}{\perp} \quad f \neq g$$

$$(E) \quad \frac{\exists x(x = t \wedge \phi)}{\phi[x/t]} \quad x \notin \mathcal{V}(t)$$

... some more rules omitted.

Quantifier Elimination

- Structure \mathcal{A} has the property of quantifier elimination:
For every formula $\exists x(a_1 \wedge \dots \wedge \neg a_n \wedge \dots)$ exists an equivalent (in \mathcal{A}) formula without variables.
- Yields a decision algorithm for the theory of \mathcal{A} (if one can decide validity of closed formulas without quantifiers).
- Used by Presburger 1929 to decide the FO theory of $\mathbb{N}, 0, 1, +, =$. (with a slight extension of the logical language).
- Used by Tarski 1930 (published '51) to decide the FO theory of $\mathbb{R}, 0, 1, +, *, =$.

Quantifier Elimination for Herbrand

Hubert Comon & Pierre Lescanne, 1988 :

- Problem: in general one can only *trade quantifiers* :
- Example : $\exists y, z x = f(y, z)$ cannot be expressed without quantifiers.
- Over Herbrand(a,f,g), it is equivalent to $x \neq a \wedge \forall y x \neq g(y)$.
- This can be used to eliminate the number of quantifier *alternations*, and finally leads to decidability of the complete FO theory of Herbrand.

What is nice about quantifier elimination:

- This is rewriting, and we have some good tools to deal with it.
- It is often challenging to *find* the right set of transformation rules ...
- ... but once we have a set of rules it often is not very difficult to check the simplification system:
 - Confluence is not important, it is sufficient that all transformations are equivalence transformations.
 - Termination: use the Rewriting toolbox of termination techniques!
 - Completeness: usually easy to check.

Complexity ? Application to other theories ?

Applications to other theories:

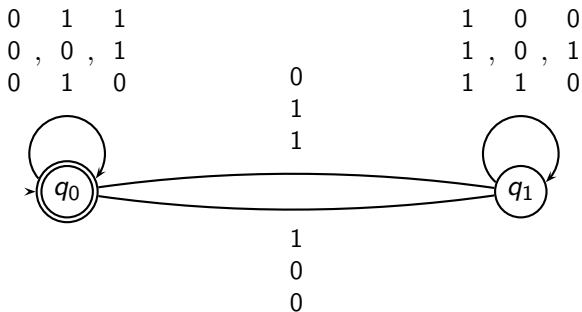
- FO-theory of Herbrand with membership constraints is decidable (Comon & Delor '90)
- FO-theory of Herbrand modulo *shallow* equational theories is decidable (Comon, Haberstrau, Jouannaud'92)

Unfortunately ...

- FO-theory of Herbrand modulo AC, or modulo A, is undecidable (Quine'46, T.'90, Marcinkowski'99)
- Even for Herbrand, the complexity is non-elementary (Vorobyov'96, Mielniczuk'04)

A completely different technique: automata!

An automaton for $x_1 = x_2 + x_3$ (in binary notation)



Example:	5	1	0	1
	= 3	1	1	0
	+ 2	0	1	0

Automatic Structures

- Idea used by Büchi '60 for WS1S, S1S, Presburger arithmetic, and by Comon & Boudet '96 for their refinement for Presburger arithmetic.
- Idea used by Thatcher and Wright '68 for deciding the theory of WS2S.
- Idea used by Rabin '69 for deciding the theory of S2S.
- First systematic investigation by Blumensath & Grädel '00.

What is an automatic structure ?

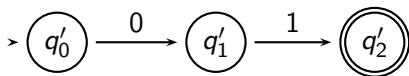
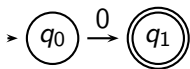
- A partial interpretation of syntactic objects (words, ...) as values in the structure (which be automatically extended to tuples of objects/values)
- An automaton recognizing the domain of the interpretation.
- For every relation symbol, an automaton recognizing the semantics of the relation (represented as tuples of syntactic objects).

What does this buy us?

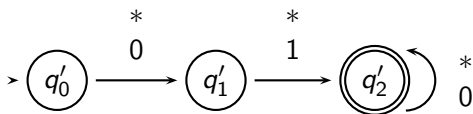
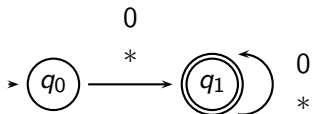
- (Classical) automata are closed under complement, union, intersection.
- Consequence: we get Boolean combinations for free!
- Attention: one still has to ensure closure under cylindrification, in order to lift two automata to the same set of variables.
- (Classical) automata are closed under projection, so we even get construction of automata for any FO formula !
- Formula valid \Leftrightarrow Accepted language nonempty.

Cylindrification

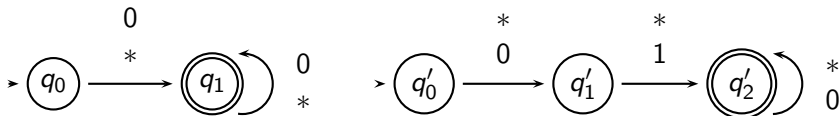
Automaton for $x_1 = 0$ and $x_2 = 2$:



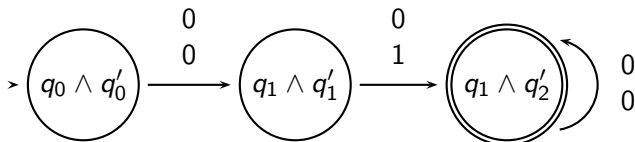
Cylindrification yields:



Product of cylindrified automata



The product now corresponds to $x_1 = 0 \wedge x_2 = 2$:



What is nice about automata ?

- One only has to construct automata for atomic formula, all the rest is for free (since classical automata have very nice properties).
- One particular useful class of automata is *Rabin* automata, used for showing decidability of $S2S$. This is a very deep and difficult result, and we should be able to benefit from this.

What is not so nice about automata ?

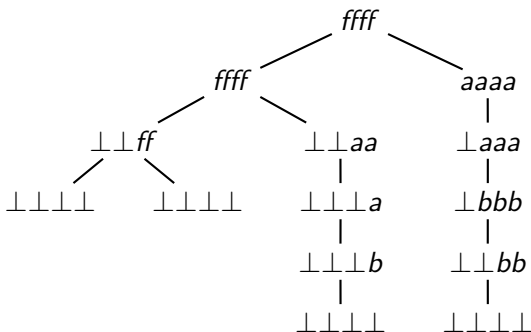
- We might need to find a new class of automata in order to express the predicates of our structure.
- This might be difficult as we need a lot of closure properties: (Boolean operations, projection, cylindrification) and decidability of emptiness.
- In general, classes of automata with constraints do not have closure under cylindrification!

Reduction automata and cylindrification

- Reduction automata (Caron, Comon, Coquidé, Dauchet, Jacquemard '94) can perform a limited number of (dis-)equality tests.
- They can be used to check whether a non-linear pattern matches a tree.
- They enjoy closure under Boolean operations, emptiness is decidable.
- But one cannot close them under cylindrification!

Automata plus component-wise tests

Emptiness undecidable even when tests at the root only (T. '01, Su '02).



PCP solution sequence $((\epsilon, \epsilon), (a, aab), (aabb, aabb))$

Automaton, plus tests at the root: $1_3 = \epsilon_1 \wedge 1_4 = \epsilon_2$

The theory of reducability

- Reduction automata have been used to show decidability of the theory of reducability (Herbrand, without equality, but with predicates “ t matches x ” for fixed t .)
- Reduction automata are not closed under cylindrification, and furthermore one cannot close them cylindrification and retain decidability.
- Isn't there a contradiction?

Reducability is a *Monadic* structure

- Monadic structure: only unary predicate symbols.
- If the language is monadic, then *any* FO formula can be rewritten into a Boolean combination of formulas of the form

$$\exists x \left(P_1(x) \wedge \dots \wedge P_n(x) \wedge \neg Q_1(x) \dots \wedge \neg Q_m(x) \right)$$

(Löwenheim '15, Ackermann '54)

- Consequences: For monadic structures, Boolean closure plus decidability of emptiness are sufficient!

How to structure our proof ?

- Simplification techniques : these can be nicely structured into hierarchical systems: one has a first system of basic simplification, then a system that operates only on normal forms w.r.t. the first system, and so on.
Termination and completeness can also benefit from such a structure.
- Automata : we do not have good techniques to bring a hierarchical structure into automata. All states are equal.

Automata are more expressive than needed

- If \mathcal{A} is automatic, then the extension of the FO theory of \mathcal{A} by the quantifier \exists^∞ is decidable (Blumensath & Grädel '00).
- Idea: $\exists^\infty x \psi(x, y)$ is true
 - iff, for given y , there are solutions for ψ such that the “length” of x is arbitrarily greater than the “length” of y .
 - iff there is solution for ψ such that the “length” of x exceeds the “length” of y by the number of states of the automaton for ψ . (Pumping Lemma !)

From finite objects to infinite objects

What happens when we wish to extend a decidability result from a class of finite objects to infinite objects ?

- Simplification : usually no problems as we are dealing with *formula*, not semantics objects directly.
For instance in case of Herbrand (finite trees to infinite trees): essentially adaption of the occur check.
- Automata : may be tough. We have to generalize automata to deal with infinite objects. Automata on infinite objects require game-theoretic acceptance conditions.
Example : from classical tree automata to Rabin automata !

Combinations of structures

Given structures $\mathcal{A}_1, \mathcal{A}_2$ over the same domain, how easily can we get decidability of $\mathcal{A}_1 + \mathcal{A}_2$ (combined structure) from decidability of \mathcal{A}_1 and decidability of \mathcal{A}_2 ?

- Simplification : there is no support for this, we have to revise each simplification procedure for the combination with the predicates from the other structure.
- Automata : if both structures use the same class of automata and the same representation of values by syntactic objects, then combination is for free!

Decidability of structures is not modular

Well-known counter-example:

- Theory of $\mathbb{N}, 0, 1, +, *, =$ is undecidable (Gödel '31)
- Presburger arithmetic ($\mathbb{N}, 0, 1, +, =$) is automatic (Büchi '60)
- Skolem arithmetic ($\mathbb{N}, 0, 1, *, =$) is automatic (Niwinski ?)

However, both Presburger and Skolem arithmetic use completely different syntactic representations of values.

Applications of combinations

- Let equational theories E_i such that each Herbrand, $=_{E_i}$ has a decidable theory (for instance : shallow theories).
- Has Herbrand, $=_{E_1}, \dots, =_{E_n}$ a decidable theory ?
- Simplification is completely lost here since the congruences are not compatible:

$$\exists x (x =_{E_1} t \wedge l_2 =_{E_2} r_2 \wedge \dots \wedge l_n \neq_{E_n} r_n \wedge \dots)$$

- In case of automatic structures (for instance when the E_i are ground) : easy!

Conclusion

- Simplification and automata seem to be inherently different.
- Automata seem unable to handle constraints like $x = f(y, z)$ (except completely new encoding of values as objects).
- Simplification/quantifier elimination seems unable to handle theories of structures like $S2S$.

Is there still hope for a Grand Unified Theory of symbolic constraint solving?