Coalgebraic up-to techniques

Damien Pous[1]*

CNRS, LIP, ENS Lyon, France
damien.pous@ens-lyon.fr

1 The concrete case of finite automata

A simple algorithm for checking language equivalence of finite automata consists in trying to compute a *bisimulation* that relates them. This is possible because language equivalence can be characterised coinductively, as the largest bisimulation.

More precisely, consider an automaton $\langle S, t, o\rangle$, where S is a (finite) set of states, $t : S \to \mathcal{P}(S)^A$ is a non-deterministic transition function, and $o : S \to 2$ is the characteristic function of the set of accepting states. Such an automation gives rise to a determinised automaton $\langle \mathcal{P}(S), t^\sharp, o^\sharp\rangle$, where $t^\sharp : \mathcal{P}(S) \to \mathcal{P}(S)^A$ and $o^\sharp : \mathcal{P}(S) \to 2$ are the natural extensions of t and o to sets. A *bisimulation* is a relation R between sets of states such that for all sets of states X, Y, $X \mathrel{R} Y$ entails:

1. $o^\sharp(X) = o^\sharp(Y)$, and
2. for all letter a, $t^\sharp_a(X) \mathrel{R} t^\sharp_a(Y)$.

The coinductive characterisation is the following one: *two sets of states recognise the same language if and only if they are related by some bisimulation.*

Taking inspiration from concurrency theory [4,5], one can improve this proof technique by weakening the second item in the definition of bisimulation: given a function f on binary relations, a *bisimulation up to* f is a relation R between states such that for all sets X, Y, $X \mathrel{R} Y$ entails:

1. $o^\sharp(X) = o^\sharp(Y)$, and
2. for all letter a, $t^\sharp_a(X) \; f(R) \; t^\sharp_a(Y)$.

For well-chosen functions f, bisimulations up to f are contained in a bisimulation, so that the improvement is sound. So is the function mapping each relation to its equivalence closure. In this particular case, one recover the standard algorithm by Hopcroft and Karp [2]: two sets can be skipped whenever they can already be related by a sequence of pairwise related states.

One can actually do more, by considering the function c mapping each relation to its congruence closure: the smallest equivalence relation which contains

* Appeared as an invited talk in Proc. CALCO'13, vol. 8089 of LNCS, pages 34-35, Springer, 2013. Work partially funded by the PiCoq and PACE projects, ANR-10-BLAN-0305 and ANR-12IS02001

the argument, and which is compatible w.r.t. set union:

$$\frac{}{X\ c(R)\ X} \qquad \frac{Y\ c(R)\ X}{X\ c(R)\ Y} \qquad \frac{X\ c(R)\ Y \qquad Y\ c(R)\ Z}{X\ c(R)\ Z}$$

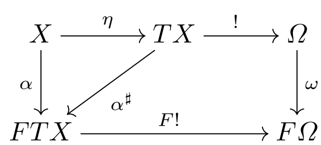
$$\frac{X\ R\ Y}{X\ c(R)\ Y} \qquad \frac{X_1\ c(R)\ Y_1 \qquad X_2\ c(R)\ Y_2}{X_1 \cup X_2\ c(R)\ Y_1 \cup Y_2}\ .$$

This is how we obtained HKC [1], an algorithm that can be exponentially faster than Hopcroft and Karp's algorithm or more recent antichain algorithms [7].

2 Generalisation to coalgebra

The above ideas generalise nicely, using the notion of λ-bialgebras [3].

Let T be a monad, F an endofunctor, and λ a distributive law $TF \Rightarrow FT$, a λ-*bialgebra* is a triple $\langle X, \alpha, \beta\rangle$, where $\langle X, \alpha\rangle$ is a F-coalgebra, $\langle X, \beta\rangle$ a T-algebra, and $\alpha \circ \beta = F\beta \circ \lambda_X \circ T\alpha$. Given such a λ-bialgebra, FT-algebra generalise non-deterministic automata: take $X \mapsto 2 \times X^A$ for F, and $X \mapsto \mathcal{P}_f X$ for T. Determinisation through the powerset construction can be generalised as follows [6], when the functor F has a final coalgebra $\langle \Omega, \omega\rangle$:



Bisimulations up-to can be expressed in a natural way in such a framework. One can in particular consider bisimulations up to congruence, where the congruence is taken w.r.t. the monad T: the fact that λ is a distributive law ensures that this improvement is always sound.

References

1. F. Bonchi and D. Pous. Checking NFA equivalence with bisimulations up to congruence. In *POPL*, pages 457–468. ACM, 2013.
2. J. E. Hopcroft and R. M. Karp. A linear algorithm for testing equivalence of finite automata. Technical Report 114, Cornell University, December 1971.
3. B. Klin. Bialgebras for structural operational semantics: An introduction. *TCS*, 412(38):5043–5069, 2011.
4. R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.
5. D. Sangiorgi. On the bisimulation proof method. *Mathematical Structures in Computer Science*, 8:447–479, 1998.
6. A. Silva, F. Bonchi, M. Bonsangue, and J. Rutten. Generalizing the powerset construction, coalgebraically. In *Proc. FSTTCS*, volume 8 of *LIPIcs*, pages 272–283. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010.
7. M. De Wulf, L. Doyen, T. A. Henzinger, and J.-F. Raskin. Antichains: A new algorithm for checking universality of finite automata. In *Proc. CAV*, volume 4144 of *Lecture Notes in Computer Science*, pages 17–30. Springer, 2006.