Accelerating transducers

Dennis Dams   Yassine Lakhnech

Martin Steffen

Bell Labs
Murray Hill, NJ

Verimag
Grenoble

Inst. für Informatik u. Prakt. Mathematik
CAU Kiel

30th of March, 2001

Friday afternoon's seminar, DoCS, U. of Chicago

Overview

- model checking and regular languages
- transducers
- iterating transducers
- conclusion

Model checking

- successful verification technique
- show that M has property φ:

$$\boxed{M \models \varphi}$$

- "push-button"
- via state exploration

$\Rightarrow$ state-space explosion problem

Model checking (cont'd)

- specifically nasty[1] instance of too big state space: infinite many states
- reasons: infinite data, infinite control (e.g. parameterized systems), time . . .
- scores of approaches:
  - use your own brain (and your own time . . .): theorem proving
  - abstraction
  - symbolic techniques (many):

    > symbolic = don't explore states one-by-one, but represent sets of states "symbolically" and explore them all at the same time

- 3 questions:
  1. how to represent infinite sets of states
  2. how to represent the reduction relation?
  3. how to calculate the reachable states in a finite amount of time?

[1] and quite common, for that matter

Regular model checking

- *very* successful finite description/symbolic representation of infinite "objects": regular languages

⇒ regular model checking (e.g., for parameterized systems $P_1 \parallel P_2 \parallel \ldots$, (cf. [7][9][1][8]. . .):

represent

- local state as letters of an alphabet
- global states as linear arrangement of local ones = word

⇒ infinite sets of states = reg. language

⇒ computation step, i.e., non-det. change of language = transduction

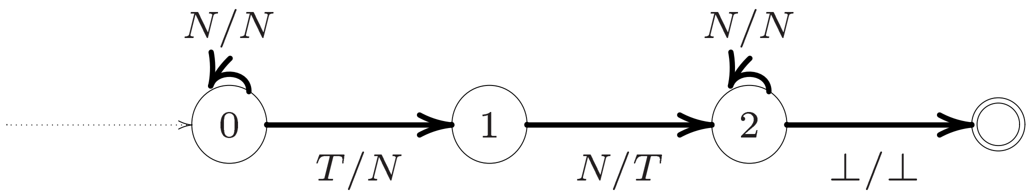
Example

Example 1. [Token array] *"Parameterized" processes: each one either has the token or not (states T and N). Token can be passed between neighbors from left to right, initially, the token is owned by the left-most process.*

Initial configuration:	TN^*
one step:	$TN \to NT$

Example (cont'd)

- Effect of one-step reduction relation: captured by a transducer



- *e.g.:* $\mathcal{T}(NTNN) = \{NNTN\}$

$\Rightarrow$ exploit for symbolic exploration: $\mathcal{T}^n \circ \mathcal{A}$

$$\begin{aligned} &= \{t' \in \mathcal{T}^n(t) \mid \text{and } t \text{ accepted by } \mathcal{A}\} \\ &= \{t' \mid t \rightarrow^n t', t \text{ accepted by } \mathcal{A}\} \end{aligned}$$

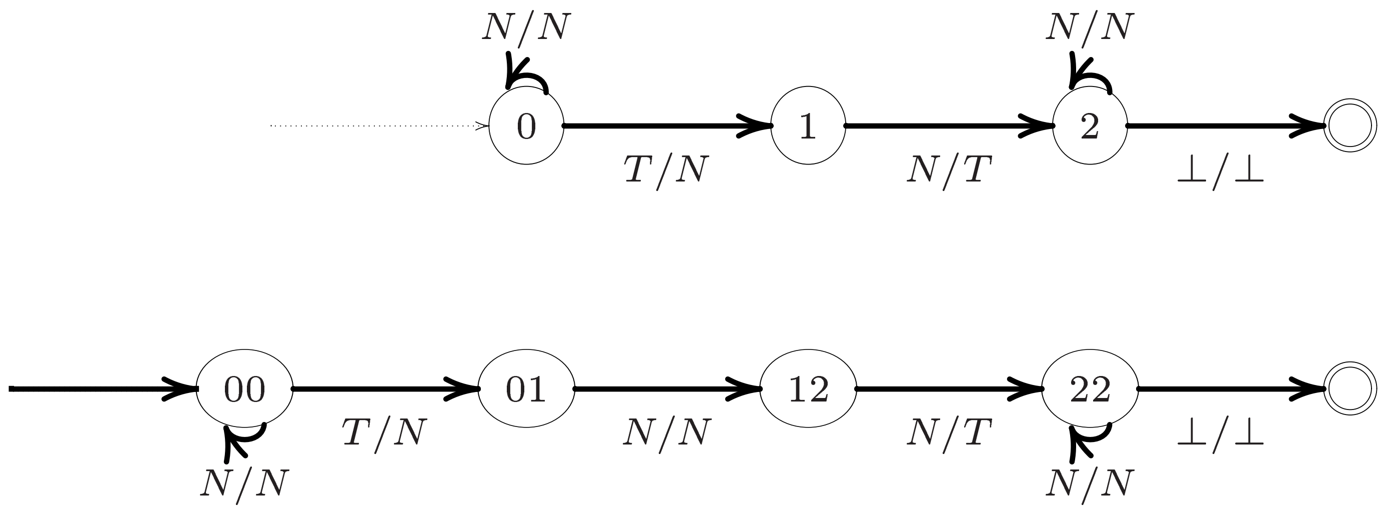
Goal: iterating transducers

- assuming that you know how to calculate $T_1 \circ \mathcal{T}_2$ by a product construction: calculate $\mathcal{T}^*$ as fixpoint $\mu X.\mathcal{T} \circ (X \cup \mathcal{T}_{id})$?, but

  1. $\mathcal{T}^*$ may not be representable as finite transducer (e.g.: duplicating the number of letter a: $q_0 a(x) \rightarrow a a q_0(x)$)

  2. even if: calculating $\mu X.\mathcal{T} \circ (X \cup \mathcal{T}_{id})$ iteratively will in general diverge following page

Example: first 2 iterations



A finite representation for $\mathcal{T}^*$?

- a sound infinite representation $\mathcal{T}^{<\omega}$ for $\mathcal{T}^*$ is straighforward (using Q^* as set of states)

$\Rightarrow$ for a finite representation: build a quotient $\mathcal{T}^{<\omega}_{/\cong}$

$\Rightarrow$ remains:

1. What to take for $\cong$?
2. How to compute $\mathcal{T}^{<\omega}_{/\cong}$?

Key observation for quotienting

Theorem 2. [Soundness] *given* $F, P \subseteq Q^*$

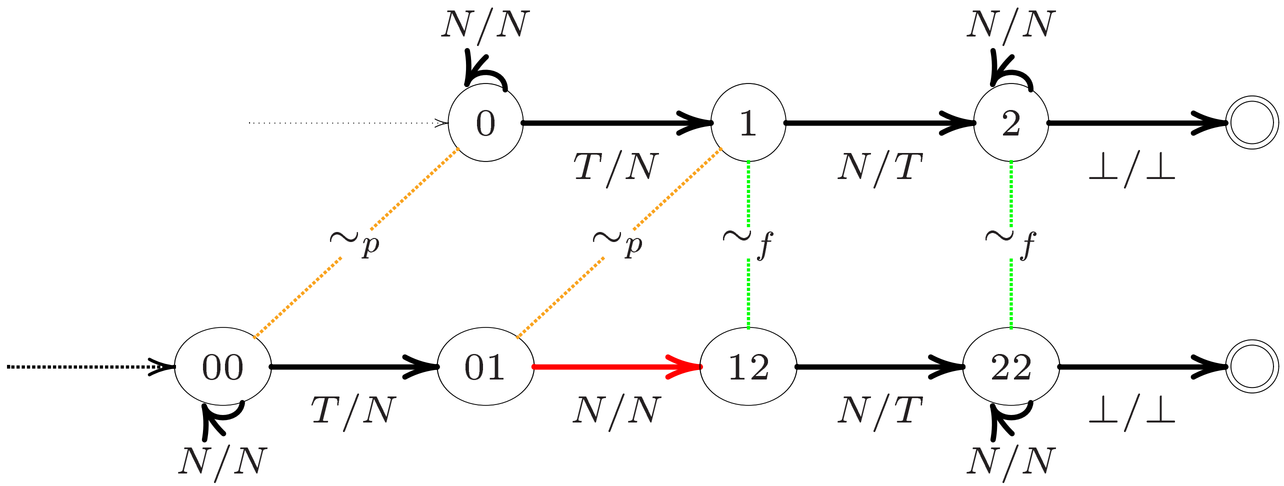
- F *and* P *two bisimulations (future and past)*
- F *and* P *swap, meaning that*

$$F; P = P; F$$

$\Rightarrow$

$$\boxed{[\![\mathcal{T}^{<\omega}]\!] = [\![\mathcal{T}^{<\omega}_{/F;P}]\!]}$$

Example, revisted



$q_{01} \sim_p q_1 \sim_f q_{12}$

But still: how to *compute* $\mathcal{T}^{<\omega}_{/F;P}$?

$\mathcal{T}^{<\omega}$ is infinite! (for Q^* is)

- way out:
  - calulate bisim's P and P on finite appoximations $\mathcal{T}^{\leq n}$
  - "extrapolate" to $\mathcal{T}^{<\omega}$

- How to extrapolate?

$\Rightarrow$ use rewriting theory, replace P and F by $\leftrightarrow^*_F$ and $\leftrightarrow^*_P$.

  - bisimulations are congruences wrt. to the monoid Q^*
  - extrapolate swapping condition (for instance): if $\leftrightarrow_P$ and $\leftrightarrow_F$ are confluent and swap, then so are $\leftrightarrow^*_P$ and $\leftrightarrow^*_F$.

$\Rightarrow$ bisimulations found in finite $\mathcal{T}^{\leq n}$ can be used to quotient $\mathcal{T}^{<\omega}$

Algorithm

```
input  𝒯 = (Q, Q_0, Σ, R)
𝒳 = 𝒯_id ;
repeat
   𝒳 := (𝒯 ∘ 𝒳) ∪ 𝒯_id ;
   determine  bisimulations   F and P on 𝒳 s.t.
      ↔_F and ↔_P swap and each possess the diamond
      property ;
until  𝒳_/≡ ∼_f (𝒯 ∘ 𝒳_/≡) ∪ 𝒯_id
```

Math form of the listing:

input $\mathcal{T} = (Q, Q_0, \Sigma, R)$

$\mathcal{X} = \mathcal{T}_{id}$;

repeat

$\mathcal{X} := (\mathcal{T} \circ \mathcal{X}) \cup \mathcal{T}_{id}$;

determine bisimulations F and P on $\mathcal{X}$ s.t. $\leftrightarrow_F$ and $\leftrightarrow_P$ swap and each possess the diamond property ;

until $\mathcal{X}_{/\equiv} \sim_f (\mathcal{T} \circ \mathcal{X}_{/\equiv}) \cup \mathcal{T}_{id}$

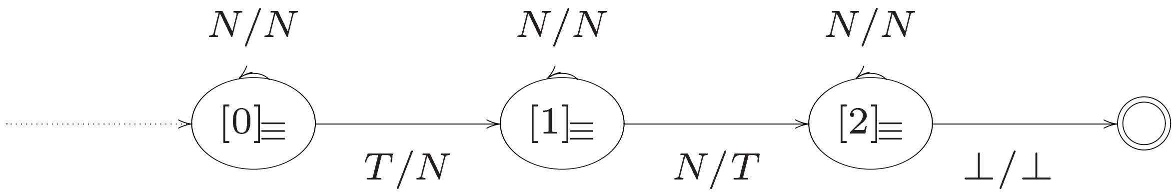
Example

- Rewrite system after 2 iterations:

$$\begin{aligned} 00 &\rightarrow 0 \\ 01 &\rightarrow 1 \\ 12 &\rightarrow 1 \\ 22 &\rightarrow 2 \end{aligned}$$

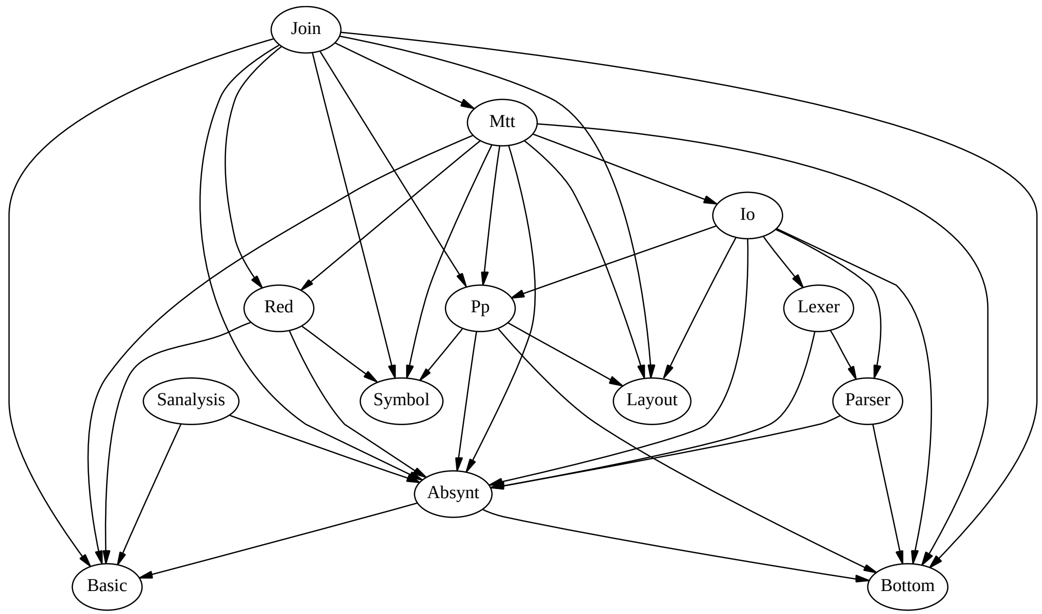
i.e.

$[0] = \{0, 00, \ldots\}$,
$[1] = \{1, 01, 001, \ldots, 12, 122, \ldots\}$,
$[2] = \{2, 22, \ldots\}$.



Implementation

- library of transducer-operations (iteration, composition, transduction)
- in *ocaml*
- efficiency: sufficient for small examples



Conclusion and further directions

- characterize iterable transducers, complexity?
- ϵ-transitions and weak bisimulation?
- Compare with
  - monadic string rewriting [3]
  - column-transducers of k-bounded depth [9]
- possible to specialize: $\mathcal{T}^{\leq n} \circ \mathcal{A}$. The construction carries over? Does one benefit from that?
- more complicated examples, dynamic process creation
- implementation: efficiency, various optimizations
- further into the jungle of tree transducers[2] . . .

[2]for tackling data, one needs trees not just words.

References

[1] Parosh Aziz Abdulla, Ahmed Bouajjani, and Bengt Jonsson. On-the-fly analysis of systems with unbounded lossy Fifo-channels. In Alan J. Hu and Moshe Y. Vardi, editors, *Proceedings of CAV '98*, volume 1427 of *Lecture Notes in Computer Science*, pages 305–318. Springer-Verlag, 1998.

[2] Parosh Aziz Abdulla, Ahmed Bouajjani, Bengt Jonsson, and Marcus Nilsson. Handling global conditions in parameterized system verification. In Nicolas Halbwachs and Doron Peled, editors, *CAV '99: Computer Aided Verification*, volume 1633 of *Lecture Notes in Computer Science*, pages 134–145. Springer-Verlag, 1999.

[3] Ronald Book and Friedrich Otto. *String Rewriting Systems*. Monographs in Computer Science. Springer-Verlag, 1993.

[4] Hubert Comon, Max Dauchet, Rémi Gilleron, Denis Lugiez, Sophie Tison, and Marc Tommasi. Tree automata, techniques and application. Available electronically.

[5] Dennis Dams, Yassine Lakhnech, and Martin Steffen. Iterating transducers for safety of abstraction. Internal Report TR-ST-00-2, Christian-Albrechts-Universität, Lehrstuhl Softwaretechnologie, May 2000.

[6] Dennis Dams, Yassine Lakhnech, and Martin Steffen. Iterating transducers. 2001. To appear.

[7] Bengt Jonsson and Marcus Nilsson. Transitive closures for regular relations for verifying infinite-state systems.

[8] Y. Kesten, O. Maler, M. Marcus, A. Pnueli, and E. Shahar. Symbolic model checking with rich assertional languages. In Orna Grumberg, editor, *CAV '97, Proceedings of the 9th International Conference on Computer-Aided Verification, Haifa. Israel*, volume 1254 of *Lecture Notes in Computer Science*. Springer, June 1997.

[9] Marcus Nilsson. Regular model checking, 2000. Licenciate Thesis of Uppsala University, Department of Information Technology.