Accelerating transducers

Dennis Dams  Yassine Lakhnech  Martin Steffen

Bell Labs  Verimag  Grenoble
Murray Hill, NJ  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 $\varphi$:

  $M \models \varphi$

- "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: $TN \rightarrow NT$
• Effect of one-step reduction relation: captured by a transducer

\[ N/N \]
\[ \text{0} \rightarrow \text{1} \rightarrow \text{2} \rightarrow \bullet \]
\[ T/N \quad N/T \quad \perp/\perp \]

• e.g.: \( \mathcal{T}(NTNN) = \{NNTN\} \)

\[ \Rightarrow \text{exploit for symbolic exploration: } \mathcal{T}^n \circ \mathcal{A} \]

\[ = \left\{ t' \in \mathcal{T}^n(t) \mid \text{and } t \text{ accepted by } \mathcal{A} \right\} \]
\[ = \left\{ t' \mid t \rightarrow^n t', t \text{ accepted by } \mathcal{A} \right\} \]
Goal: iterating transducers

- assuming that you know how to calculate $T_1 \circ T_2$ by a product construction: calculate $T^*$ as fixpoint $\mu X. T \circ (X \cup T_{id})$, but

1. $T^*$ 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 $\mu X. T \circ (X \cup T_{id})$ 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 straightforward (using $Q^*$ as set of states)

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

$\Rightarrow$ remains:

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

Theorem 2. [Soundness]  \(\text{given } F, P \subseteq Q^*\)

- \(F\) and \(P\) two bisimulations (future and past)

- \(F\) and \(P\) swap, meaning that

\[
F; P = P; F
\]

\(\Rightarrow\)

\[[T^<\omega] = [T^<\omega]/_{F; P} \]
Example, revisited

\[ q_{01} \sim_p q_1 \sim_f q_{12} \]
But still: how to compute $\mathcal{T}_{/F;P}^{<\omega}$?

$\mathcal{T}_{/F;P}^{<\omega}$ is infinite! (for $Q^*$ is)

- way out:
  - calculate bisim's $P$ and $P$ on finite approximations $\mathcal{T}_{/F;P}^{\leq n}$
  - "extrapolate" to $\mathcal{T}_{/F;P}^{<\omega}$

- How to extrapolate?

  $\Rightarrow$ use rewriting theory, replace $P$ and $F$ by $\leftrightarrow_P^*$ and $\leftrightarrow_F^*$.

  - 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 $\mathcal{T}_{/F;P}^{\leq n}$ can be used to quotient $\mathcal{T}_{/F;P}^{<\omega}$
Algorithm

\begin{algorithm}
\caption{Algorithm for Bisimulation Congruence}
\begin{algorithmic}
\Require $\mathcal{T} = (Q, Q_0, \Sigma, R)$
\State $\mathcal{X} = \mathcal{T}_{id};$
\Repeat
\State $\mathcal{X} := (\mathcal{T} \circ \mathcal{X}) \cup \mathcal{T}_{id};$
\State determine bisimulations $F$ and $P$ on $\mathcal{X}$ s.t. $\leftrightarrow_F$ and $\leftrightarrow_P$ swap and each possess the diamond property;
\Until $\mathcal{X}_{/\equiv} \sim_f (\mathcal{T} \circ \mathcal{X}_{/\equiv}) \cup \mathcal{T}_{id}$
\end{algorithmic}
\end{algorithm}
Example

- Rewrite system after 2 iterations:

\[
\begin{align*}
00 & \rightarrow 0 \\
01 & \rightarrow 1 \\
12 & \rightarrow 1 \\
22 & \rightarrow 2
\end{align*}
\]

i.e.

\[
\begin{align*}
[0] &= \{0, 00, \ldots\}, \\
[1] &= \{1, 01, 001, \ldots, 12, 122, \ldots\}, \\
[2] &= \{2, 22, \ldots\}.
\end{align*}
\]
Implementation

- library of transducer-operations (iteration, composition, transduction)
- in ocaml
- efficiency: sufficient for small examples
Conclusion and further directions

- characterize iterable transducers, complexity?
- $\epsilon$-transitions and weak bisimulation?
- Compare with
  - monadic string rewriting [3]
  - column-transducers of $k$-bounded depth [9]
- possible to specialize: $T^{\leq n} \circ A$. 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


