Accelerating transducers

Dennis Dams    Yassine Lakhnech
Martin Steffen

Dept. of Electrical Engineering
TU Eindhoven

Verimag
Grenoble

Inst. für Informatik u. Prakt. Mathematik
CAU Kiel

June 2000
Vires-Review, Autrans
Overview

- Safety of data-abstractions
- Transducers
- Iterating transducers
- Conclusion
Regular languages and transducers

- **regular model checking** for parameterized systems (cf. Pnueli’s talk yesterday):
  
  represent
  - locations as letters of an alphabet
  - global states as word
  - reachable states as *reg. languages*
  - computation step as *transduction*

- regularity in data: unbounded *integers*, *queue content*: cf. **LASH**
Model checking & data

- data domains in general infinite ($\mathbb{N}$, lists, ...)

$\Rightarrow$ brute state exploration fails

$\Rightarrow$ - data independance
  - symbolic exploration
  - theorem proving
  - abstraction techniques
Data abstraction

- 2 data domains:
  - concrete $D$: infinite
  - abstract $D^\alpha$: finite

- operations

\[
\begin{align*}
f & : D \to D \\
\alpha & : D^\alpha \to D^\alpha
\end{align*}
\]

- abstraction $\alpha : D \to D^\alpha$

Safety of abstraction

\[
\begin{array}{ccc}
D^\alpha & \xrightarrow{\alpha} & D^\alpha \\
\downarrow & & \downarrow \\
D & \xrightarrow{f} & D
\end{array}
\]

or, more general: $\alpha(f(x)) \in f^\alpha(\alpha(x))$
Safety of data abstraction

Given concrete and abstract system $S$ and $S^\alpha$:

\[
\text{Safety of abstraction} \quad S^\alpha \models \varphi' \implies S \models \varphi
\]

- $D$, $D^\alpha$, $f$, $\alpha$: given inductively $\Rightarrow$ standard proof-technique: induction over $D$

- here: exploiting the finiteness of $D^\alpha$
  - represent the inductively given functions as rewrite rules
  - invert the direction of the rules.

$\Rightarrow$ acceleration needed.
Example 1. [Parity] concrete domain $\mathbb{N}$, abstract domain $D^\alpha = \text{even} \mid \text{odd}$. Functions $\alpha$, $f$, and $f^\alpha$: inductively given, here as rewrite system

\[
\begin{align*}
\alpha(0) & \rightarrow \text{even} & \neg \text{even} & \rightarrow \text{odd} & f x & \rightarrow \text{succ}^2 x \\
\alpha(\text{succ } x) & \rightarrow \neg (\alpha x) & \neg \text{odd} & \rightarrow \text{even} & f^\alpha x & \rightarrow x
\end{align*}
\]

\[
\begin{array}{ccccccccc}
even & \leftarrow & \neg \text{odd} & \leftarrow & \neg^2 \text{even} & \leftarrow & \neg^3 \text{odd} & \leftarrow & \neg^4 \text{even} & \leftarrow & \cdots \\
\alpha 0 & \uparrow & \neg^2 (\alpha 0) & \uparrow & \neg^4 (\alpha 0) & \cdots
\end{array}
\]

\[
\begin{array}{ccccccccc}
\ast & \uparrow & \alpha (\text{succ}^2 0) & \ast & \uparrow & \alpha (\text{succ}^4 0)
\end{array}
\]
Representation

• Questions:
  1. how to represent infinite sets of words (terms)?
  2. how to represent the reduction relation?
  3. how to calculate the reachable states in a finite amount of time?

• representation
  1. infinite sets by regular languages/automata
  2. reduction steps by transducer

• definition of transducer: cf. the example
Example (cont’d)

\[\begin{align*}
\alpha(0) & \rightarrow \text{even} & \neg \text{even} & \rightarrow \text{odd} & f x & \rightarrow \text{succ}^2 x \\
\alpha(\text{succ } x) & \rightarrow \neg (\alpha x) & \neg \text{odd} & \rightarrow \text{even} & f^\alpha x & \rightarrow x
\end{align*}\]

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

- e.g.: \(T(\neg (\alpha(\text{succ}(0)))) = \{\neg^2 \alpha(0)\}\)

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

\[= \{t \mid T^n(t) \text{ accepted by } A\}\]

\[= \{t \mid t \rightarrow_{n}^{n} t', t' \text{ accepted by } A\}\]
Composing transducers

- regular (word) transductions are closed under composition

- Notation: $\mathcal{T}_2 \circ \mathcal{T}_1$

- product construction

\[
\begin{align*}
q_j(f(x)) & \rightarrow t' & q_i(t') & \rightarrow^* t \\
q_{i,j}(f(x)) & \rightarrow t
\end{align*}
\]  

where $t' \in T_\Sigma(Q(X))$ and $t \in T_\Sigma(Q^2(X))$. 

(COMP)
Iterating transducers

- 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 the letter $a$: $q_0a(x) \rightarrow aaq_0(x)$)

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

![Diagram of a transducer]

- Calculations:
  - From state 0: $\alpha/\epsilon$, $\neg/\neg$, $0/\text{even}/0$, $\text{succ}/\neg\alpha$
  - From state 1: $\alpha/\epsilon$, $\text{succ}/\text{succ}$
  - From state 2: $\alpha/\epsilon$, $\text{succ}/\text{succ}$

- Transitions:
  - From state 00: $\neg/\neg$, $\text{succ}/\neg\alpha$, $0/\text{even}$
  - From state 01: $\neg/\neg$, $0/\text{even}$
  - From state 22: $0/0$
A finite representation for $\mathcal{T}^*$?

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

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

$\Rightarrow$ remains:

1. What to take for $\sim$?

2. How to compute $\mathcal{T}^{\leq \omega}_{/\sim}$?
Algorithm: idea

- exploiting
  - associativity of composition
  - congruence of composition\(^1\)

\[ \Rightarrow \text{equivalence between states can be turned into a} \]
  - strongly normalizing and
  - confluent

  ground rewrite system on \( Q^* \)

- represent equivalence class \( q[\alpha] \) by normal form \( q_{\text{norm}}(\alpha) \)

\(^1\)i.e., if \( q_\alpha \cong q_{\alpha'} \) then \( q_{\beta\alpha\gamma} \cong q_{\beta\alpha'\gamma} \)
Key observation for quotienting

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

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

- $F$ and $P$ swap

$\Rightarrow$

$$[T \leq \omega] = [\mathcal{T}^{\leq \omega}]$$

$^2F; P = P; F$
Example: $\mathcal{T}_\alpha$

$q_0 \sim_p q_1 \sim_f q_{12}$
Algorithm

input \( \mathcal{T} = (Q, Q_0, \Sigma, R) \);
repeat
\( \mathcal{T}^{\leq n+1} = \mathcal{T} \circ \mathcal{T}^{\leq n} \cup \mathcal{T}_{id} \);
determine bisimulations \( F \) and \( P \) on \( \mathcal{T}^{\leq n+1} \) with \( F \) and \( P \) confluent and commuting;
\( \mathcal{T}^* = \mathcal{T}^{\leq n+1}_{/F;P} \);
until \( \mathcal{T} \circ \mathcal{T}^* = \mathcal{T}^* \)

Efficiency

- combinatorial state space explosion

\( \Rightarrow \) necessary to avoid costly

1. recomputation of reductions
2. recomputation equivalence of states of \( \mathcal{T} \)
Example: $\mathcal{T}_\alpha$

- 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

- implementation
  - efficiency
  - incorporating states of arities $> 1$ (macro tree transducers)
  - front-end

- characterizing iterable transducers

- more complicated examples
References


