Verification of parametrised shared-memory asynchronous systems

Igor Walukiewicz
CNRS, Bordeaux University
IAS, TU Munich

Joint work with: M. Fortin, S. LaTorre, A. Muscholl
Model: asynchronous, shared memory

Hague, 2011

Esparza, Ganty, Majumdar, 2013
Durand-Gasselin, Esparza, Ganty, Majumdar, 2015

leader

Reg

contributors
Model: asynchronous, shared memory

Hague, 2011

Esparza, Ganty, Majumdar, 2013

Durand-Gasselin, Esparza, Ganty, Majumdar, 2015

Register: lock-free reads/writes

contributors
Leader and contributors are pushdown processes.

- If there is only one contributor: leader+contributor can simulate a Turing machine.
- For unknown number of contributors the model becomes surprisingly manageable.
Semantics

\[ C = \langle S, \delta \subseteq S \times \Sigma_C \times S, s_{init} \rangle \quad D = \langle T, \Delta \subseteq T \times \Sigma_D \times T, t_{init} \rangle. \]

\( G \): a finite set of register values

A configuration is \((M, t, g)\), where \(M \in \mathbb{N}^S\), \(t \in T\), \(g \in G\).

\[(M, t, g) \xrightarrow{w(h)} (M, t', h) \quad \text{if } t \xrightarrow{w(h)} t' \text{ in } \Delta,\]

\[(M, t, g) \xrightarrow{r(h)} (M, t', h) \quad \text{if } t \xrightarrow{r(h)} t' \text{ in } \Delta \text{ and } h = g,\]

\[(M, t, g) \xrightarrow{\bar{w}(h)} (M', t, h) \quad \text{if } M \xrightarrow{\bar{w}(h)} M' \text{ in } \delta,\]

\[(M, t, g) \xrightarrow{\bar{r}(h)} (M', t, h) \quad \text{if } M \xrightarrow{\bar{r}(h)} M' \text{ in } \delta \text{ and } h = g.\]

where

\( M \xrightarrow{a} M' \text{ in } \delta \quad \text{if } s \xrightarrow{a} s' \text{ in } \delta \text{ and } M' = M - [s] + [s'], \text{ for some } s, s' \in S.\)
Given a leader $D$ from some class of systems $G$ and a configuration $C$, if

$$M \xrightarrow{w(h)} (M, t', h)$$

then $M \xrightarrow{w(h)} (M', t, h)$ if $M \xrightarrow{\bar{w}(h)} M'$ in $\delta$, and

$$M \xrightarrow{\bar{r}(h)} (M', t, h)$$

if $M \xrightarrow{\bar{r}(h)} M'$ in $\delta$ and $h = g$.

where

$$M \xrightarrow{a} M' \text{ in } \delta$$

if $s \xrightarrow{a} s' \text{ in } \delta$ and $M' = M - [s] + [s']$, for some $s, s' \in S$. 

\[ 
\begin{array}{cccc}
  t & \xrightarrow{a} & t' & t' \\
  g & & g' & g'' \\
  s_1 & & s_1' & s_1' \\
  \vdots & & \vdots & \vdots \\
  s_i & & s_i' & s_i' \\
  \vdots & & \vdots & \vdots \\
  s_n & & s_n' & s_n' \\
\end{array} \]
Set semantics

\[(B, t, g) \xrightarrow{w(h)} (B, t', h) \quad \text{if } t \xrightarrow{w(h)} t' \text{ in } \Delta,\]

\[(B, t, g) \xrightarrow{r(h)} (B, t', h) \quad \text{if } t \xrightarrow{r(h)} t' \text{ in } \Delta \text{ and } h = g,\]

\[(B, t, g) \xrightarrow{w(h)} (B', t, h) \quad \text{if } B \xrightarrow{w(h)} B' \text{ in } \delta,\]

\[(B, t, g) \xrightarrow{\bar{r}(h)} (B', t, h) \quad \text{if } B \xrightarrow{\bar{r}(h)} B' \text{ in } \delta \text{ and } h = g.\]

\[B \xrightarrow{a} B' \text{ in } \delta \quad \text{if } s \xrightarrow{a} s' \text{ in } \delta \text{ and } B' = B \cup \{s'\}, \text{ for some } s, s' \in S.\]

\[(M, t, g) \xrightarrow{w(h)} (M, t', h) \quad \text{if } t \xrightarrow{w(h)} t' \text{ in } \Delta,\]

\[(M, t, g) \xrightarrow{r(h)} (M, t', h) \quad \text{if } t \xrightarrow{r(h)} t' \text{ in } \Delta \text{ and } h = g,\]

\[(M, t, g) \xrightarrow{w(h)} (M', t, h) \quad \text{if } M \xrightarrow{w(h)} M' \text{ in } \delta,\]

\[(M, t, g) \xrightarrow{\bar{r}(h)} (M', t, h) \quad \text{if } M \xrightarrow{\bar{r}(h)} M' \text{ in } \delta \text{ and } h = g.\]

\[M \xrightarrow{a} M' \text{ in } \delta \quad \text{if } s \xrightarrow{a} s' \text{ in } \delta \text{ and } M' = M - [s] + [s'], \text{ for some } s, s' \in S.\]
Multiset

\[ t \xrightarrow{a} t' \quad t' \quad t' \]

\[ g \rightarrow g' \rightarrow g'' \rightarrow g''' \]

\[ s_1 \quad s_1 \xrightarrow{b} s_1' \quad s_1' \]
\[ \vdots \quad \vdots \quad \vdots \quad \vdots \]
\[ s_i \quad s_i \quad s_i \xrightarrow{c} s_i' \quad s_i' \]
\[ \vdots \quad \vdots \quad \vdots \quad \vdots \]
\[ s_n \quad s_n \quad s_n \quad s_n \]

Set

\[ t \xrightarrow{a} t' \quad t' \quad t' \]

\[ g \rightarrow g' \rightarrow g'' \rightarrow g''' \]

\[ s_1 \quad s_1 \quad s_1' \quad s_1' \]
\[ \vdots \quad \vdots \quad \vdots \quad \vdots \]
\[ s_i \quad s_i \xrightarrow{b} s_i \quad s_i' \quad s_i' \]
\[ \vdots \quad \vdots \quad \vdots \quad \vdots \]
\[ s_n \quad s_n \quad s_n \xrightarrow{c} s_i' \quad s_i' \]
\[ \vdots \quad \vdots \quad \vdots \quad \vdots \]
\[ s_1' \quad s_1' \quad s_i' \quad s_i' \]
C,D may be infinite state

\[ C = \langle S, \delta \subseteq S \times \Sigma_C \times S, s_{init} \rangle \quad D = \langle T, \Delta \subseteq T \times \Sigma_D \times T, t_{init} \rangle. \]

\[
\begin{aligned}
(M, t, g) \xrightarrow{w(h)} & (M, t', h) \quad \text{if } t \xrightarrow{w(h)} t' \text{ in } \Delta, \\
(M, t, g) \xrightarrow{r(h)} & (M, t', h) \quad \text{if } t \xrightarrow{r(h)} t' \text{ in } \Delta \text{ and } h = g, \\
(M, t, g) \xrightarrow{\bar{w}(h)} & (M', t, h) \quad \text{if } M \xrightarrow{\bar{w}(h)} M' \text{ in } \delta, \\
(M, t, g) \xrightarrow{\bar{r}(h)} & (M', t, h) \quad \text{if } M \xrightarrow{\bar{r}(h)} M' \text{ in } \delta \text{ and } h = g.
\end{aligned}
\]

Transition systems C and D need not to be finite. In our case they are given by pushdown systems:

\[
\begin{aligned}
\mathcal{A}_C = \langle P, \Sigma_C, \Gamma_C, \delta, p_{init}, A^C_{init} \rangle \quad \mathcal{A}_D = \langle Q, \Sigma_D, \Gamma_D, \Delta, q_{init}, A^D_{init} \rangle.
\end{aligned}
\]

So \( S = \{ q\alpha : q \in P, \alpha \in \Gamma^*_C \} \)
Every contributor proposes a value
Leader chooses one of these values
The rest of the protocol uses the chosen value

Example of a system:

Leader eventually decides on a value
If the leader decides on the value, contributors use only this value.
On runs where only one value is used i.o. the protocol is correct
Example of a system:

- Contributors proposes values.
- Leader chooses one of these values.
- The rest of the protocol uses the chosen value.

Example properties:
(for every n, for every run)

- Leader eventually decides on a value
- If the leader decides on the value, contributors use only this value.
- On runs where only one value is used i.o. the protocol is correct

There is a run where the leader has decided on some value and afterwards a contributor is using a different value.
Example of a system:
- Contributors proposes values.
- Leader chooses one of these values.
- The rest of the protocol uses the chosen value.

Example properties:
(for every n, for every run)
- Leader eventually decides on a value
- If the leader decides on the value, contributors use only this value.
- On runs where only one value is used i.o. the protocol is correct.

There is a maximal run where the leader does not decide on a value.
Example of a system:

* Contributors proposes values.
* Leader chooses one of these values.
* The rest of the protocol uses the chosen value.

Example properties:
(for every n, for every run)

* Leader eventually decides on a value
* If the leader decides on the value, contributors use only this value.
* On runs where only one value is used i.o. the protocol is correct

safe run
reachability
more general liveness property
(C,D)-systems

- Reachability
- Repeated reachability
- Safety
- Verification of properties

Properties

- PSPACE-complete for pushdowns
- Almost always decidable
- Hierarchical (C,D)-systems
- PSPACE-complete for pushdowns
- NEXPTIME-complete for pushdowns
- PSPACE-complete if only about infinite runs
We are interested in the complexity of deciding these properties when C, D are pushdown systems.
(C,D)-systems

- Reachability
  - PSPACE-complete for pushdowns
  - Almost always decidable
  - Hierarchical (C,D)-systems

- Repeated reachability

- Safety

- Verification of properties
  - Nexpctime-complete
Reachability in (C,D)-systems

Given a leader D from some class of systems $\mathcal{D}$ and a contributor $C$ from some class $\mathcal{C}$, is there some value $n$ such that $D||C||\ldots||C$ ($n$-times) have a run writing some particular value into the register?

Fact

When $C$ and $D$ are the class of pushdown systems and $n$ is fixed then the problem is undecidable.
Reachability in (C,D)-systems

Given a leader D from some class of systems $\mathcal{D}$ and a contributor $C$ from some class $\mathcal{C}$, is there some value $n$ such that $D||C||\ldots||C$ ($n$-times) have a run writing some particular value into the register?

Thm [Hague, Esparza et al.]

When $C$ and $D$ are the class of pushdown systems then the reachability problem is decidable, and PSPACE-complete.
Thm
Let $C$ and $D$ be both effectively closed under synchronised product with finite automata.
If $C$ has decidable reachability problem and $D$ has effective downward closure, then reachability for $(C,D)$-systems is decidable.
Let $\mathcal{C}$ and $\mathcal{D}$ be both effectively closed under synchronized product with finite automata.
If $\mathcal{C}$ has decidable reachability problem and $\mathcal{D}$ has effective downward closure, then reachability for $(\mathcal{C}, \mathcal{D})$-systems is decidable.

$\mathcal{C}$ is effectively closed under synchronized product with finite automata:
given M from $\mathcal{C}$ and a finite automaton A, the synchronized product of M and A belongs to $\mathcal{C}$ and can be effectively constructed.

$\mathcal{D}$ has effective downward closure:
given M from $\mathcal{D}$, the finite automaton accepting all (scattered) subwords of traces of M can be constructed effectively.
Effective downward closure:

- pushdown automata [Courcelle 1991]
- Petri nets [Habermehl et al. 2010]
- stacked counter automata [Zetzsche 2015]
- higher-order pushdown with collapse automata [Clemente, Parys, Salvati, W. 2016]

Theorem applies to

leader: pushdown automata, Petri nets, decidable subclasses of multi-stack, stacked counter automata.

contributors: any of the above, lossy channel systems, hierarchical composition of (C,D)-systems.
Hierarchical composition of (C,D)-systems

leader $P_0$ and each subtree (C,D)-system is contributor
(C,D)-systems

- Reachability
  - PSPACE-complete for pushdowns
  - Almost always decidable
  - Hierarchical (C,D)-systems

- Repeated reachability
  - PSPACE-complete for pushdowns

- Safety

- Verification of properties
Repeated reachability in (C,D)-systems

Given a leader $D$ from some class of systems $\mathcal{D}$ and contributors $C_1, \ldots, C_n, \ldots$ from some class $\mathcal{C}$, is there some value $n$ such that $D \parallel C_1 \parallel \cdots C_n$ write some particular value into the register infinitely often?

**Thm** [Durand-Gasselin, Esparza, Ganty, Majumdar 2015]

When $\mathcal{C}$ and $\mathcal{D}$ are the class of pushdown systems then the liveness problem is decidable is PSPACE-hard and in NEXPTIME.
Liveness in (C,D)-systems

Given a leader $D$ from some class of systems $\mathcal{D}$ and contributors $C_1, \ldots, C_n, \ldots$ from some class $\mathcal{C}$, is there some value $n$ such that $D \parallel C_1 \parallel \cdots C_n$ write some particular value into the register infinitely often?

Thm

When $\mathcal{C}$ and $\mathcal{D}$ are the class of pushdown systems then the liveness problem is decidable is PSPACE-complete.
(C,D)-systems

- Reachability
  - Almost always decidable
  - Hierarchical (C,D)-systems

- Repeated reachability
  - PSPACE-complete for pushdowns

- Safety
  - NEXPTIME-complete for pushdowns
  - PSPACE-complete if only about infinite runs

- Verification of properties
Safety in (C,D)-systems

Given a leader $D$ from some class of systems $\mathcal{D}$ and contributors $C_1, \ldots, C_n, \ldots$ from some class $\mathcal{C}$, is there some value $n$ such that $D \parallel C_1 \parallel \cdots \parallel C_n$ has a maximal run that does not write some particular value into the register?

Thm

When $\mathcal{C}$ and $\mathcal{D}$ are the class of pushdown systems then the safety problem is NEXPTIME-complete.
Thm
When \( C \) and \( D \) are the class of pushdown systems then the safety problem is NEXPTIME-complete.

Thm
Let \( C \) and \( D \) be the class of pushdown systems.
Knowing if there is some infinite safe run in PSPACE-complete.
Knowing if there is some maximal finite safe run in NEXPTIME-complete.

Thm
If \( C \) is a class of finite systems and \( D \) be the class of pushdown systems then the problems are coNP-complete.
Prop

When \( \mathcal{C} \) and \( \mathcal{D} \) are the class of pushdown systems then the existence of a maximal finite safe run is \text{NEXPTIME-hard}.

Reduction of a tiling problem:
Find a tiling with letters from \( \Sigma \) of a \( 2^n \times 2^n \) square.
The tiling should respect neighbourhood relations \( H, V \subseteq \Sigma \times \Sigma \).

Leader writes: \( A_{1,1}, \overline{A_{1,1}}, A_{1,2}, \overline{A_{1,2}}, \ldots, A_{1,2^n}, \overline{A_{1,2^n}}, \ldots, A_{2^n,2^n}, \overline{A_{2^n,2^n}} (\$\$)^{2^n} \).

and checks the horizontal dependencies.
**Prop**

When \( C \) and \( D \) are the class of pushdown systems then the existence of a maximal finite safe run is NEXPTIME-hard.

Reduction of a tiling problem:

Find a tiling with letters from \( \Sigma \) of a \( 2^n \times 2^n \) square.

The tiling should respect neighbourhood relations \( H, V \subseteq \Sigma \times \Sigma \).

Leader writes:

\[
A_{1,1}, \overline{A_{1,1}}, A_{1,2}, \overline{A_{1,2}}, \ldots, A_{1,2^n}, \overline{A_{1,2^n}}, \ldots, A_{2n,2}, \overline{A_{2n,2}}, \ldots, A_{2n,2^n}, \overline{A_{2n,2^n}} (\$\$)^{2^n} \cdot
\]

and checks the horizontal dependencies.

Contributors check vertical dependencies.

We can ensure that contributors
- read all the symbols, and
- every vertical dependency is checked by some contributor.
(C,D)-systems

- Reachability
  - Almost always decidable
  - Hierarchical (C,D)-systems
- Repeated reachability
  - PSPACE-complete for pushdowns
- Safety
  - NEXPTIME-complete for pushdowns
  - PSPACE-complete if only about infinite runs
- Verification of properties
  - C-stutter-expanding properties
  - NEXPTIME-complete
A trace is a sequence of register operations during a run. A maximal trace comes from a maximal run (finite or infinite).

A property of traces is \( P \subseteq (\Sigma_D \cup \Sigma_C)^\infty \).
A property is C-stutter-expanding if it is closed under duplicating actions of contributors.

\[
\text{If } x \bar{w}(g) y \in P \text{ then } x \bar{w}(g)\bar{w}(g) y \in P
\]

Verification of properties of (C,D)-systems
Given a C-stutter-expanding property \( P \). Given a leader \( D \) from some class of systems \( D \) and contributors \( C_1, \ldots, C_n, \ldots \) from some class \( C \), is there some value \( n \) such that \( D \parallel C_1 \parallel \cdots C_n \) has a maximal trace in \( P \).
Verification of properties of (C,D)-systems

Given a C-stutter-expanding property $P$. Given a leader $D$ from some class of systems $\mathcal{D}$ and contributors $C_1, \ldots, C_n, \ldots$ from some class $\mathcal{C}$, is there some value $n$ such that $D \parallel C_1 \parallel \cdots C_n$ has a maximal trace in $P$.

All previously considered properties are special instances:

- reachability: $P$ is the set of traces containing the special action.
- repeated reachability: $P$ is the set of traces containing the special action infinitely often.
- safety: $P$ is the set of traces without the special action.
Verification of properties of (C,D)-systems

For a Buchi automaton for C-stutter-expanding property P. Given a leader D from some class of systems $\mathcal{D}$ and contributors $C_1, \ldots, C_n, \ldots$ from some class $\mathcal{C}$, is there some value $n$ such that $D \parallel C_1 \parallel \cdots \parallel C_n$ has a maximal trace in P.

Verification for arbitrary regular properties is undecidable, as with a property we can require that there is only one copy of a contributor.

A property of traces is $P \subseteq (\Sigma_{\mathcal{D}} \cup \Sigma_{\mathcal{C}})^\infty$.
A property is C-stutter-expanding if it is closed under duplicating actions of contributors.

If $x \bar{w}(g) y \in P$ then $x \bar{w}(g)\bar{w}(g) y \in P$.

Thm

When $\mathcal{C}$ and $\mathcal{D}$ are the class of pushdown systems then verification of properties of (C,D)-systems is decidable and NEXPTIME-complete.
(C,D)-systems of pushdown process have very good algorithmic properties

- Verification of C-stutter-expanding properties is decidable in NEXPTIME
- For some relevant subclasses it is PSPACE.

The NEXPTIME-hardness argument shows that they can exhibit quite a nontrivial behaviour.

Changing from one two arbitrary many contributors turns the problem from undecidable to manageable.