Multi-Core Model Checking

Alfons Laarman
November 14, 2013
State Space Explosion

An exponential problem

- system data
- system components
- property size

(LaQuSo project)
State Space Explosion

An exponential problem

- system data
- system components
- property size

Approach

- multi-core model checking
An exponential problem

- system data
- system components
- property size

System data

Approach

- multi-core model checking
- Confluence / partial-order reduction
- Symbolic techniques (BDD-based and SAT-based)
- On-the-fly techniques
- Compression techniques
### Research questions

- Can model checking scale (linearly, ideally) on modern multi-cores?

#### Speedup:

\[ S_P = \frac{T_{seq}}{T_P} \]

- Ideal: \( S_P = P \)
- Linear: \( S_P = P/c \)
Research questions

- Can model checking scale (linearly, ideally) on modern multi-cores?
  - Formalisms: plain, timed, stochastic, etc
  - Properties: Reachability, LTL, CTL, etc

Speedup:

\[ S_P = \frac{T_{seq}}{T_P} \]

Ideal: \( S_P = P \)

Linear:

\[ S_P = \frac{P}{c} \]
Research questions

- Can model checking scale (linearly, ideally) on modern multi-cores?
  - Formalisms: plain, timed, stochastic, etc
  - Properties: Reachability, LTL, CTL, etc
- Are our parallel solutions compatible with other techniques?

Speedup:
\[ S_P = \frac{T_{\text{seq}}}{T_P} \]
Ideal: \( S_P = P \)
Linear: \( S_P = \frac{P}{c} \)

- Partial-order reduction (POR)
- Symbolic exploration
- On-the-fly techniques
- Compression techniques
Challenges

**Difficulties of parallelism**

- Correctness of data structures and algorithms
Challenges

Difficulties of parallelism

- Correctness of data structures and algorithms
- Steep memory hierarchies
Challenges

Difficulties of parallelism

- Correctness of data structures and algorithms
- Steep memory hierarchies
- Cache coherence protocol

```c
#define B (1024 * 1024 * 1024)

int main (void) {
    int result = 0;
    for (int i = 0; i < B; i++)
        result++;
    return result;
}
```
Challenges

Difficulties of parallelism

- Correctness of data structures and algorithms
- Steep memory hierarchies
- Cache coherence protocol

```c
#define B (1024*1024*1024)

int main (void) {
    int result = 0;
    for (int i = 0; i < B; i++)
        result++;
    return result;
}

#define P 16

static void count (void *arg) {
    int *counter = (int *) arg;
    for (int i = 0; i < B / P; i++)
        (*counter)++;
}

int main (void) {
    pthread_t thread[P];
    int counters[P] = 0;
    for (int i = 0; i < P; i++)
        pthread_create (&thread[i], NULL, count, &counters[i]);

    int result = 0;
    for (int i = 0; i < P; i++)
        result += counters[i];

    return result;
}
```
**Challenges**

**Difficulties of parallelism**

- Correctness of data structures and algorithms
- Steep memory hierarchies
- Cache coherence protocol

```c
#define B (1024*1024*1024)   T = 27

int main (void) {
    int result = 0;
    for (int i = 0; i < B; i++)
        result++;
    return result;
}
```

```c
#define P 16

static void count (void *arg) {
    int *counter = (int *) arg;
    for (int i = 0; i < B / P; i++) (*counter)++;
}

int main (void) {
    pthread_t thread[P];
    int counters[P] = 0;

    for (int i = 0; i < P; i++)
        pthread_create (&thread[i], NULL, count, &counters[i]);

    int result = 0;
    for (int i = 0; i < P; i++)
        pthread_join (thread[i], NULL);
    result += counters[i];

    return result;
}
```

\[ T_{16} = 32 \]
Challenges

Difficulties of parallelism

- Correctness of data structures and algorithms
- Steep memory hierarchies
- Cache coherence protocol (false sharing)

```c
#define B (1024*1024*1024)  T = 27

int main (void) {
    int result = 0;
    for (int i = 0; i < B; i++)
        result++;
    return result;
}
```

```c
#define P 16
static void count (void *arg) {
    int *counter = (int *) arg;
    for (int i = 0; i < B / P; i++) (*counter)++;
}

int main (void) {
    pthread_t thread[P];
    int __attribute__((aligned(64))) counters[P] = 0;
    for (int i = 0; i < P; i++)
        pthread_create (&thread[i], NULL, count, &counters[i]);

    int result = 0;
    for (int i = 0; i < P; i++)
        result += counters[i];
    return result;
}
```

\[ T_{16} = 32 \]

\[ T_{16} = 1.8 \]
(Explicit-state) reachability

Problem:
find all reachable states from $s_0 \in S$ using a next-state function: $\text{post}(S) \rightarrow 2^S$

A state $s \in S$ is a (fixed) $K$-sized vector: $\langle v_1, \ldots, v_K \rangle$
Static partitioning or shared hash table

- **Static partitioning**
  - On-the-fly (BFS)
  - ± Scalability (communication on queues)

- **Shared hash table**
  - On-the-fly: (pseudo) DFS & BFS
  - ± Scalability
Static partitioning or shared hash table

Static partitioning

- On-the-fly (BFS)
- Scalability (communication on queues)
**Static partitioning or shared hash table**

**Static partitioning**
- X On-the-fly (BFS)
- ± Scalability (communication on queues)

**Shared hash table**
- ✓ On-the-fly: (pseudo) DFS & BFS
- ? Scalability
Main bottlenecks

- State store: concurrent access
- Graph traversal: Random memory access (bandwidth)
Lockless Hash Table: Design
Laarman, van de Pol, Weber [fmcad10]

Main bottlenecks
- State store: concurrent access
- Graph traversal: Random memory access (bandwidth)

Design
- Hash memoization
- Walking the Line
- In-situ locking
Experiments from 2010 (BEEM database)

- SPIN 5.2.4 (NASA/JPL)
- DiVinE 2.2 (Brno,CZ)
- LTSmin (shared hash table)
Experiments from 2010 (BEEM database)

Impact

- SPIN model checker ................. [HOLZMANN’12]
- GPU model checking ...... [SULEWSKI ET AL ’11,12]
- Parallel BDDs ....... VAN DIJK, LAARMAN, VAN DE POL
  [AVOCS12][PDMC12]
### Reachability

- Scalability comes from limiting bandwidth usage
- Correctness established with model checker

<table>
<thead>
<tr>
<th>Explicit state</th>
<th>Compression</th>
<th>POR</th>
<th>On-the-fly</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>

Reachability | ✓ | ? | ? | ✓
Scalability comes from limiting bandwidth usage
Correctness established with model checker

- Partial-order reduction can be computed (state) locally
Reachability

- Scalability comes from limiting bandwidth usage
- Correctness established with model checker

<table>
<thead>
<tr>
<th>Explicit state</th>
<th>Compression</th>
<th>POR</th>
<th>On-the-fly</th>
<th>Reachability</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

- Partial-order reduction can be computed (state) locally
- No compression, but states are often very similar due to locality

\[
\langle 3, 5, 5, 4, 1, 3 \rangle \rightarrow \langle 3, 5, 9, 3, 1, 3 \rangle
\]
Recursive indexing

\[ H_K \]

\((K - 1) \times H_2\)
**Recursive indexing**

[HOLZMANN 97][BLOM ET AL. 08]

\[ H_K \]

\[ (K - 1) \times H_2 \]

![Recursive indexing diagram](image)

- Combinatorial \( \implies \) balanced tree \((N + 2\sqrt{N} + 4^{\sqrt{(N)}}) \cdots \approx N)\)
- Compresses states of length K to almost 2!

\[ H_K \]

\[ (K - 1) \times H_2 \]

\[ \sqrt{N} \]

\[ \sqrt{N} \]

\[ \text{Combinatorial} \implies \text{balanced tree} \]

\[ \text{Compresses states of length K to almost 2!} \]
Recursive indexing

[HOLZMANN 97][BLOM ET AL. 08]

\[ H_K \]

\[ (K - 1) \times H_2 \]

\[ \sqrt{N} \]

\[ \sqrt{N} \]

\[ N \]

- Combinatorial \( \implies \) balanced tree \((N + 2\sqrt{N} + 4\sqrt[4]{N}) \cdots \approx N)\)

- Compresses states of length K to almost 2!

- Hard to parallelize (flatliners)
Solution

- Temporary binary tree structure on stack
Solution

- Temporary binary tree structure on stack

\[
\langle 3, 5, 5, 4, 1, 3 \rangle
\]

\[
\langle 3, 5, 5 \rangle \quad \langle 4, 1, 3 \rangle
\]

\[
\langle 3, 5 \rangle \quad \langle 3, 5 \rangle \quad \langle 4, 1 \rangle
\]
### Solution

- Temporary binary tree structure on stack
- Reuse lockless hash table (merge tables)
Parallel Tree Compression
Laarman, van de Pol, Weber [spin11]

Solution

- Temporary binary tree structure on stack
- Reuse lockless hash table (merge tables)
**Solution**

- Temporary binary tree structure on stack
- Reuse lockless hash table (merge tables)
Solution

- Temporary binary tree structure on stack
- Reuse lockless hash table (merge tables)
- Incremental updates: \((K - 1) \rightarrow \log_2(K - 1)\) lookups

\[ \langle 3, 5, 5, 4, 1, 3 \rangle \rightarrow \langle 3, 5, 9, 4, 1, 3 \rangle \]
Experiments from 2011 [BEEM database]
Laarman, van de Pol, Weber [spin11]
Experiments from 2011 [BEEM database]
Laarman, van de Pol, Weber [spin11]

Information theoretical lower bound

> View states as stream of variables:

\[
\langle v_1^1, ..., v_K^1 \rangle, \langle v_1^2, ..., v_K^2 \rangle, ...
\]

with

\[
|v_i^j| = 2^{32}
\]

> \( p(v_i^j = v_{i-1}^j) = K - 1 \)

> \( p(v_i^j, v_{i-1}^j) = 1 \) (under-estimation)

> Entropy per state:

\[
K \times H(s_i^j) \approx \log_2(2^{32}) + \log_2(K) \text{ bits} \approx 1 + \epsilon \text{ integer}
\]

> Halve the root table with Cleary compact hash table [memics11]
Experiments from 2011 [BEEM database]
Laarman, van de Pol, Weber [spin11]

Information theoretical lower bound?

- View states as stream of variables: \( \langle v_1^1, \ldots v_K^1 \rangle, \langle v_1^2, \ldots v_K^2 \rangle, \ldots \) with \( |v_j^i| = 2^{32} \)
Experiments from 2011 [BEEM database]
Laarman, van de Pol, Weber [spin11]

Information theoretical lower bound?

- View states as stream of variables: \( \langle v_1, \ldots, v_K \rangle, \langle v_2, \ldots, v_K \rangle, \ldots \) with \( |v_j| = 2^{32} \)
- \( p(v_j = v_j^{-1}) = \frac{K-1}{K} \) and \( p(v_j \neq v_j^{-1}) = \frac{1}{K} \) (under-estimation)
Experiments from 2011 [BEEM database]
Laarman, van de Pol, Weber [spin11]

Information theoretical lower bound?

- View states as stream of variables: \(<v_1^1,\ldots,v_K^1>,<v_1^2,\ldots,v_K^2>,\ldots\) with \(|v_j^i|=2^{32}\)
- \(p(v_j^i=v_j^{i-1})=\frac{K-1}{K}\) and \(p(v_j^i\neq v_j^{i-1})=\frac{1}{K}\) (under-estimation)
- Entropy per state: \(K \times H(s_j^i) \approx \log_2(2^{32}) + \log_2(K)\) bits \(\approx 1 + \epsilon\) integer

\[\langle 3,5,5,4,1,3 \rangle \xrightarrow{\frac{K-1}{K}} \langle 3,5,9,4,1,3 \rangle \xrightarrow{\frac{1}{K}} \langle 3,5,9,3,2,3 \rangle\]
Experiments from 2011 [BEEM database]
Laarman, van de Pol, Weber [spin11]

Information theoretical lower bound?

- View states as stream of variables: \( \langle v_1^1, \ldots, v_1^K \rangle, \langle v_2^2, \ldots, v_2^K \rangle, \ldots \), with \( |v_j^i| = 2^{32} \)
- \( p(v_j^i = v_j^{i-1}) = \frac{K-1}{K} \) and \( p(v_j^i \neq v_j^{i-1}) = \frac{1}{K} \) (under-estimation)
- Entropy per state: \( K \times H(s_j^i) \approx \log_2(2^{32}) + \log_2(K) \) bits \( \approx 1 + \epsilon \) integer
- Halve the root table with Cleary compact hash table [MEMICS11]
Reachability

- Scalability from merging tables & incremental updates
- Correctness proved by hand
  - The recursive tree function is an injection [SPIN11]
Reachability

- Scalability from merging tables & incremental updates
- Correctness proved by hand
  - The recursive tree function is an injection [SPIN11]

Still only safety…
LTL

The $\omega$-language of the Büchi automaton represents all counter examples

[VARDI ET WOLPER 86]
The $\omega$-language of the Büchi automaton represents all counter examples

\[ \text{[Vardi et Wolper 86]} \]

```
\begin{tikzpicture}
  \node (s1) at (1, 2) [circle, draw] {2};
  \node (s2) at (1, 0) [circle, draw] {3};
  \node (s3) at (2, 0) [circle, draw] {4};
  \node (s4) at (2, 2) [circle, draw] {1};
  \node (s5) at (3, 0) [circle, draw] {5};
  \node (s6) at (3, 2) [circle, draw] {6};

  \draw[->] (s1) -- (s2);
  \draw[->] (s2) -- (s3);
  \draw[->] (s3) -- (s4);
  \draw[->] (s4) -- (s5);
  \draw[->] (s5) -- (s6);
  \draw[->] (s6) -- (s1);

\end{tikzpicture}
```

“It is as yet an open problem how a liveness verification algorithm could be generalized to the use of more than two processing cores while retaining a low search complexity.”

\[ \text{[Holzmann '07]} \]

“One of the most important open problems of parallel LTL model checking is to design an on-the-fly scalable parallel algorithm with linear time complexity.”

\[ \text{[Brim, Barnat et Ročkai '11]} \]
Nested Depth-First Search for LTL

[COURCOUNETIS’93]

procedure DFSblue(s)
    s.cyan := true
    for all s’ in post(s) do
        if ¬t.blue ∧ ¬t.cyan then
            DFSblue(s’)
        if accepting(s) then
            DFSred(s)
    s.blue := true
    s.cyan := false

procedure DFSred(s)
    s.red := true
    for all s’ ∈ post(s) do
        if t.cyan then ExitCycle
        if ¬t.red then DFSred(s’)

Nested DFS (NDFS)

- Linear time
Nested Depth-First Search for LTL
[Courcoubetis’93]

procedure DFSblue(s)
    s.cyan := true
    for all s’ in post(s) do
        if \neg t.blue \land \neg t.cyan then
            DFSblue(s’)
        if accepting(s) then
            DFSred(s)
    s.blue := true
    s.cyan := false

procedure DFSred(s)
    s.red := true
    for all s’ in post(s) do
        if t.cyan then ExitCycle
        if \neg t.red then DFSred(s’)

Nested DFS (NDFS)
▶ Linear time
▶ DFS itself is likely not parallelizable
▶ DFS order is a P-complete problem
▶ We assume: P ≠ NC
Multi-core Nested Depth-First Search (Principle)
[ATVA11], [PDMC11], [ATVA12]

code for worker $p$:

**procedure** DFSblue($s$, $p$)

s.cyan[$p$] := true

for all $s'$ in shuffle(post($s$)) do

  if $\neg s'.blue \land \neg t.cyan[p]$ then

    DFSblue($s'$, $p$)

  if accepting($s$) then

    DFSred($s$, $p$)

  s.blue := true

  s.cyan[$p$] := false

**procedure** DFSred($s$, $p$)

s.red[$p$] := true

for all $s' \in$ post($s$) do

  if t.cyan[$p$] then ExitCycle

  if $\neg t.red[p]$ then DFSred($s'$, $p$)
Multi-core Nested Depth-First Search (Principle)

[ATVA11], [PDMC11], [ATVA12]

code for worker $p$:

```plaintext
procedure DFSblue(s,p)
    s.cyan[p] := true
    for all $s'$ in shuffle(post(s)) do
        if ¬s'.blue ∧ ¬t.cyan[p] then DFSblue(s',p)
    if accepting(s) then DFSred(s,p)
    s.blue := true
    s.cyan[p] := false

procedure DFSred(s,p)
    s.red[p] := true
    for all $s' \in post(s)$ do
        if t.cyan[p] then ExitCycle
        if ¬t.red[p] then DFSred(s',p)
```

In reality more synchronization!

Lemma 4: Blue states have blue or cyan successors:

Blue ⊆ ⋃ $p □ (Blue ∪ Cyan_p)$.
Multi-core Nested Depth-First Search (Principle)
[ATVA11], [PDMC11], [ATVA12]

code for worker $p$:

```plaintext
procedure DFSblue(s,p)
    s.cyan[p] := true
    for all s' in shuffle(post(s)) do
        if ¬s'.blue ∧ ¬t.cyan[p] then
            DFSblue(s',p)
    if accepting(s) then
        DFSred(s,p)
    s.blue := true
    s.cyan[p] := false

procedure DFSred(s,p)
    s.red[p] := true
    for all s' ∈ post(s) do
        if t.cyan[p] then ExitCycle
        if ¬t.red[p] then DFSred(s',p)
```

- In reality more synchronization!
- Laarman, Wijs et al. [ATVA11]
  Laarman et van de Pol [PDMC11]
  Evangelista, Laarman et al. [ATVA12]
Multi-core Nested Depth-First Search (Principle)

\[ [\text{ATVA11}], [\text{PDMC11}], [\text{ATVA12}] \]

code for worker \( p \):

```
procedure DFSblue(s,p)
    s.cyan[p] := true
    for all s' in shuffle(post(s)) do
        if \( \neg s'.blue \land \neg t.cyan[p] \) then
            DFSblue(s',p)
        if accepting(s) then
            DFSred(s,p)
    s.blue := true
    s.cyan[p] := false

procedure DFSred(s,p)
    s.red[p] := true
    for all s' \in post(s) do
        if t.cyan[p] then ExitCycle
        if \( \neg t.red[p] \) then DFSred(s',p)
```

- In reality more synchronization!
- Laarman, Wijs et al. \[\text{ATVA11}\]
  Laarman et van de Pol \[\text{PDMC11}\]
  Evangelista, Laarman et al. \[\text{ATVA12}\]

 Lemma 4: Blue states have blue or cyan successors:
\[
Blue \subseteq \bigcup_{p} \Box (Blue \cup Cyan_{p}).
\]
LTL and Partial-Order Reduction

- Scalability due to hash/tree table \textit{(linear-time)}
- Correctness proved by hand [ATVA11][PDMC11][ATVA12]
Scalability due to hash/tree table (linear-time)

Correctness proved by hand [ATVA11][PDMC11][ATVA12]

For partial-order reduction, we need to solve ignoring
LTL and Partial-Order Reduction

- Scalability due to hash/tree table (**linear-time**)
- Correctness proved by hand [ATVA11][PDMC11][ATVA12]

For partial-order reduction, we need to solve **ignoring**

- For **livelocks** (⊇ LTL), any unfair cycle is a counter example!
- Parallel DFS\textsubscript{FIFO} Laarman et Faragó [NFM13]
Experiments: LTL with Partial-Order Reduction

- **cndfs**:
  - garp
  - giop2.nomig
  - i-protocol2
  - leader5

- **dfs fifo**:
  - garp
  - giop2.nomig
  - i-protocol2
  - leader5

- **Max. model size explored in 30 min.**
  - LTSmin
  - DiVinE
  - cores
  - dfs fifo
  - owcty

- **dfs fifo vs owcty + POR**
  - [Brim et al '10]
Experiments: LTL with Partial-Order Reduction

Partial-order reductions:

<table>
<thead>
<tr>
<th></th>
<th>LTSmin</th>
<th>SPIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>leader</td>
<td>0.49%</td>
<td>1.15%</td>
</tr>
<tr>
<td>garp</td>
<td>2.18%</td>
<td>12.73%</td>
</tr>
<tr>
<td>giop</td>
<td>1.86%</td>
<td>2.42%</td>
</tr>
<tr>
<td>i-prot</td>
<td>31.83%</td>
<td>41.37%</td>
</tr>
</tbody>
</table>
Experiments: LTL with Partial-Order Reduction

Partial-order reductions:

<table>
<thead>
<tr>
<th></th>
<th>LTSmin</th>
<th>SPIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>leader</td>
<td>0.49%</td>
<td>1.15%</td>
</tr>
<tr>
<td>garp</td>
<td>2.18%</td>
<td>12.73%</td>
</tr>
<tr>
<td>giop</td>
<td>1.86%</td>
<td>2.42%</td>
</tr>
<tr>
<td>i-prot</td>
<td>31.83%</td>
<td>41.37%</td>
</tr>
</tbody>
</table>

Max. model size explored in 30 min.

<table>
<thead>
<tr>
<th>cores</th>
<th>LTSmin DFSFIFO</th>
<th>DiVinE OWCTY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12</td>
<td>9</td>
</tr>
<tr>
<td>48</td>
<td>15</td>
<td>11</td>
</tr>
</tbody>
</table>

DFSFIFO vs OWCTY + POR [Brim et al ’10]
<table>
<thead>
<tr>
<th>Formalism</th>
<th>Property</th>
<th>Explicit state</th>
<th>Compression</th>
<th>POR</th>
<th>On-the-fly</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain</td>
<td>Reachability</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>LTL</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>. . . . Livelocks</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Formalism</td>
<td>Property</td>
<td>Explicit state</td>
<td>Compression</td>
<td>POR</td>
<td>On-the-fly</td>
</tr>
<tr>
<td>-----------</td>
<td>----------</td>
<td>----------------</td>
<td>-------------</td>
<td>-----</td>
<td>------------</td>
</tr>
<tr>
<td><strong>Plain</strong></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>Reachability</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>LTL</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>.... Livelocks</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Timed</strong></td>
<td></td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td>Reachability</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td>LTL</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Timed Automata

States are semi-symbolic: $s = \langle d, \sigma \rangle$ (finite continuous-time abstraction)

$Z_1 := y - x \leq 0 \land y \leq 2$

$Z_2 := Z_3 := y - x = 0 \land y \leq 2$
Timed Automata

States are semi-symbolic: \( s = \langle d, \sigma \rangle \) (finite continuous-time abstraction)

\[
Z_1 := y - x \leq 0 \land y \leq 2
\]

\[
Z_2 := Z_3 := y - x = 0 \land y \leq 2
\]

This introduces a new subsumption relation: \( s \sqsubseteq s' \), iff \( d = d' \land \sigma \sqsubseteq \sigma' \)
Timed Automata

States are semi-symbolic: $s = \langle d, \sigma \rangle$ (finite continuous-time abstraction)

This introduces a new subsumption relation: $s \sqsubseteq s'$, iff $d = d' \land \sigma \sqsubseteq \sigma'$

Subsumption is a simulation relation which allows another, dynamic abstraction
Timed Automata

✓ For reachability, we implemented a lockless multi-map [FORMAT12]
For reachability, we implemented a lockless multi-map [format12]
Timed Automata

For reachability, we implemented a lockless multi-map [FORMAT12]

Subsumption does not preserve Büchi emptiness! [TRIPAKIS’09]
Analysis of accepting cycles/spirals with subsumption

Laarman, Olesen, Dalsgaard, Larsen, van de Pol [cav13]

**Lemma:** If $s$ has an accepting cycle then any $s' \sqsubseteq s$ has it as well.
Lemma: If $s$ has an accepting cycle then any $s' \sqsubseteq s$ has it as well

$$s' \sqsubseteq s$$

Preservation of accepting cycles

$s' \sqsubseteq s$

Proof Sketch

$$s \rightarrow^* t \rightarrow^+ t$$
Analysis of accepting cycles/spirals with subsumption
Laarman, Olesen, Dalsgaard, Larsen, van de Pol [cav13]

Lemma: If $s$ has an accepting cycle then any $s' \sqsubseteq s$ has it as well

Preservation of accepting cycles

<table>
<thead>
<tr>
<th>$s'$</th>
<th>$t'$</th>
<th>$t''$</th>
</tr>
</thead>
<tbody>
<tr>
<td>→*</td>
<td>→+</td>
<td></td>
</tr>
<tr>
<td>□</td>
<td>□</td>
<td>□</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$s$</th>
<th>$t$</th>
<th>$t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>→*</td>
<td>→+</td>
<td></td>
</tr>
</tbody>
</table>
Lemma: If $s$ has an accepting cycle then any $s' \sqsubseteq s$ has it as well.

Preservation of accepting cycles

<table>
<thead>
<tr>
<th>$s'$</th>
<th>$\rightarrow^*$</th>
<th>$t'$</th>
<th>$\rightarrow^+$</th>
<th>$t''$</th>
<th>$\rightarrow^+$</th>
<th>$\cdots$</th>
<th>$\rightarrow^+$</th>
<th>$t'''$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\square$</td>
<td></td>
<td>$\square$</td>
<td></td>
<td>$\square$</td>
<td>$\square$</td>
<td>$\square$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$s$</th>
<th>$\rightarrow^*$</th>
<th>$t$</th>
<th>$\rightarrow^+$</th>
<th>$t$</th>
<th>$\rightarrow^+$</th>
<th>$\cdots$</th>
<th>$\rightarrow^+$</th>
<th>$t$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\square$</td>
<td></td>
<td>$\square$</td>
<td></td>
<td>$\square$</td>
<td>$\square$</td>
<td>$\square$</td>
<td></td>
</tr>
</tbody>
</table>
Lemma: If $s$ has an accepting cycle then any $s' \sqsubseteq s$ has it as well

Preservation of accepting cycles

<table>
<thead>
<tr>
<th>$s'$ →*</th>
<th>$t'$ →+</th>
<th>$t''$ →+</th>
<th>⋯×⋯→+</th>
<th>$t'''$ →+</th>
<th>$x$</th>
</tr>
</thead>
<tbody>
<tr>
<td>⊦</td>
<td>⊦</td>
<td>⊦</td>
<td>⊦</td>
<td>⊦</td>
<td>⊦</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$s$ →*</th>
<th>$t$ →+</th>
<th>$t$ →+</th>
<th>⋯⋯⋯→+</th>
<th>$t$ →+</th>
<th>$t$</th>
</tr>
</thead>
</table>
**Lemma:** If $s$ has an accepting cycle then any $s' \sqsupseteq s$ has it as well

```
s' \rightarrow^* t' \rightarrow^+ t'' \rightarrow^+ \ldots \rightarrow^+ t'' \rightarrow^+ x
\sqcup \quad \sqcup \quad \sqcup \quad \sqcup \quad \sqcup \quad \sqcup 

\sqcup \quad \sqcup 

s \rightarrow^* t \rightarrow^+ t \rightarrow^+ \ldots \ldots \rightarrow^+ t \rightarrow^+ t
```

**Lemma:** If $t'$ has an accepting spiral then $t'$ has an accepting cycle
Results with Parallel Timed Reachability / LTL
Laarman, Olesen, Dalsgaard, Larsen, van de Pol [cav13][formats2012]

- Add full LTL to timed automata
- Runtimes 60x faster than UPPAAL on 48 cores
- Up to 70x reductions due to subsumption
- Tree compression for large discrete states
LTSmin
LTSmin Blom, van de Pol, Weber [cav09]

http://fmt.cs.utwente.nl/tools/ltsmin/ (open source)

Guard-based POR . . . . . . . PATER, LAARMAN, van de Pol [SPIN13]

promela formalism . . . . . . . . van der Berg et Laarmann [PDMC12]

LTSmin tool . . . . . . . . . . LAARMAN, WEBER, van de Pol [NFM11]
## Contributions

<table>
<thead>
<tr>
<th>Formalism</th>
<th>Property</th>
<th>Explicit state</th>
<th>Compression</th>
<th>POR</th>
<th>On-the-fly</th>
<th>publications</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Plain</strong></td>
<td>Reachability</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>[FMCAD10][SPIN11][MEMICS11]</td>
</tr>
<tr>
<td></td>
<td>LTL</td>
<td>✓</td>
<td>✓</td>
<td>1/2</td>
<td>✓</td>
<td>[ATVA11][PDMC11][ATVA12]</td>
</tr>
<tr>
<td></td>
<td>. . . Livelocks</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>[SPIN13][NFM13]</td>
</tr>
<tr>
<td><strong>Timed</strong></td>
<td>Reachability</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>[FORMATS12]</td>
</tr>
<tr>
<td></td>
<td>LTL</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>[CAV13]</td>
</tr>
</tbody>
</table>
## Contributions

<table>
<thead>
<tr>
<th>Formalism</th>
<th>Property</th>
<th>Explicit state</th>
<th>Compression</th>
<th>POR</th>
<th>On-the-fly</th>
<th>Symbolic</th>
<th>publications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain</td>
<td>Reachability</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>[FMCAD10][SPIN11][MEMICS11]</td>
</tr>
<tr>
<td></td>
<td>LTL</td>
<td>✓</td>
<td>✓</td>
<td>1/2</td>
<td>✓</td>
<td>?</td>
<td>[ATVA11][PDMC11][ATVA12]</td>
</tr>
<tr>
<td></td>
<td>. . . Livelocks</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>?</td>
<td>[SPIN13][NFM13]</td>
</tr>
<tr>
<td>Timed</td>
<td>Reachability</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>?</td>
<td>[FORMATS12]</td>
</tr>
<tr>
<td></td>
<td>LTL</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>?</td>
<td>[CAV13]</td>
</tr>
</tbody>
</table>

### Other work
- Multi-core BDDs ........................................... van Dijk, Laarman, van de Pol [PDMC12]
## Contributions

<table>
<thead>
<tr>
<th>Formalism</th>
<th>Property</th>
<th>Explicit state</th>
<th>Compression</th>
<th>POR</th>
<th>On-the-fly</th>
<th>Symbolic</th>
<th>publications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain</td>
<td>Reachability</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>[FMCAD10][SPIN11][MEMICS11]</td>
</tr>
<tr>
<td></td>
<td>LTL</td>
<td>✓</td>
<td>✓</td>
<td>1/2</td>
<td>✓</td>
<td>?</td>
<td>[ATVA11][PDLC11][ATVA12]</td>
</tr>
<tr>
<td></td>
<td>. . . Livelocks</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>?</td>
<td>[SPIN13][NFM13]</td>
</tr>
<tr>
<td>Timed</td>
<td>Reachability</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>?</td>
<td>[FORMATS12]</td>
</tr>
<tr>
<td></td>
<td>LTL</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>?</td>
<td>[CAV13]</td>
</tr>
</tbody>
</table>

### Other work

- Multi-core BDDs ......................... van Dijk, Laarman, van de Pol [PDMC12]
- One-Way-Catch-Them Young (LTL) ................ [BARNAT, BRIM, ROČKAĬ’01]
- Topological sort proviso (POR) .................. [BARNAT, BRIM, ROČKAĬ’10]
- CTL ..................................... [SAAD ET AL’12]
## Future work

<table>
<thead>
<tr>
<th>Formalism</th>
<th>Property</th>
<th>Explicit state</th>
<th>Compression</th>
<th>POR</th>
<th>On-the-fly</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain</td>
<td>Reachability</td>
<td>✔ ✔ ✔ ✔ ✔</td>
<td>✔ ✔ ✔ ✔ ✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LTL</td>
<td>✔ ✔ 1/2 ✔</td>
<td>✔ ✔ ✔ ✔ ✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Timed</td>
<td>Reachability</td>
<td>✔ ✔ – ✔</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LTL</td>
<td>✔ ✔ – ✔</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Future work

<table>
<thead>
<tr>
<th>Formalism</th>
<th>Property</th>
<th>Explicit state</th>
<th>Compression</th>
<th>POR</th>
<th>On-the-fly</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Plain</strong></td>
<td>Reachability</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>LTL</td>
<td>✓</td>
<td>✓ 1/2</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>CTL</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td><strong>Timed</strong></td>
<td>Reachability</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>LTL</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>CTL</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
## Future work

<table>
<thead>
<tr>
<th>Formalism</th>
<th>Property</th>
<th>Explicit state</th>
<th>Compression</th>
<th>POR</th>
<th>On-the-fly</th>
<th>Symbolic</th>
<th>Distributed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain</td>
<td>Reachability</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td>LTL</td>
<td>✓</td>
<td>✓</td>
<td>1/2</td>
<td>✓</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Timed</td>
<td>Reachability</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td></td>
<td>LTL</td>
<td>✓</td>
<td>✓</td>
<td>–</td>
<td>✓</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
## Future work

<table>
<thead>
<tr>
<th>Formalism</th>
<th>Property</th>
<th>Explicit state</th>
<th>Compression</th>
<th>POR</th>
<th>On-the-fly</th>
<th>Symbolic</th>
<th>Distributed</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Plain</strong></td>
<td>Reachability</td>
<td>✓  ✓  ✓  ✓  ✓  ✓</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LTL</td>
<td>✓  ✓  1/2 ✓</td>
<td>✓</td>
<td>?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Timed</strong></td>
<td>Reachability</td>
<td>✓  ✓ – ✓</td>
<td>?</td>
<td>?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LTL</td>
<td>✓  ✓ – ✓</td>
<td>?</td>
<td>?</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Future work

<table>
<thead>
<tr>
<th>Formalism</th>
<th>Property</th>
<th>Explicit state</th>
<th>Compression</th>
<th>POR</th>
<th>On-the-fly</th>
<th>Symbolic</th>
<th>Distributed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain</td>
<td>Reachability</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>?</td>
</tr>
<tr>
<td>LTL</td>
<td></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>

### Other questions

- Can our parallel DFS-based solutions be generalized?
- (Bottom-)SCC detection
- Emptiness of \{Tree, Rabin, Streett\} automata, etc.
- What search-order property is preserved?
Multi-core Nested Depth-First Search (Principle)  

[$\text{atva11}$] [$\text{pdmc11}$] [$\text{atva12}$]  

Contributions

- Recursive indexing
- Multi-core BDDs
- Combinatorial optimizations

Related work

- Information theoretical optimum?
- $\log(N)$ bits per state
- $\mathcal{O}(N)$ time
- Compresses states of length $K$ to almost $2^K$
- Hard to parallelize (flatlining)

Conclusions

- Better scalability
- View states as $K$-periodic stream of $2^K$-valued variables
- Information entropy per state: $\log_2(2^K) + \log_2(K)$ bits
- Half root table with compact hash table [$\text{memics11}$]

Experiments from 2011 [BEEM database]

- [BEEM database]
- [EMC11]
- [MEMICS11]
- [ATVA11]
- [FMCAD11]
- [ATVA12]
- [MEMICS12]
- [FMCAD13(1)]
- [MFCS13]
- [CAV13]
- [FMCAD10]
- [SPIN11]
- [MEMICS11]
- [ATVA11]
- [FMCAD11]
- [ATVA12]
- [SPIN13]
- [NFM13]
- [FORMATS12]
- [CAV13]
- [Saad et al. 12]