5. Symbolic Traversal Techniques

Content

5.1 Set-operations with OBDD's
5.2 Representation of transition systems
5.3 Symbolic state-space traversal
5.4 Equivalence of sequential circuits
5.5 Efficient forward traversal
General Problem: determine the reachable states of a sequential circuit

- Example: part of a digital filter
- Not all of the $2^{20} = 1,048,576$ states are reachable
- Have to consider all input sequences!

Why is this problem important?

- If we knew all reachable states of a system we could check if a safety-critical state is possible
- e.g., if all traffic lights were green ...
But who in the world designs systems with unreachable states ???

Sometimes we have no choice ...

![Symbolic Traversal Techniques Diagram]

- start=0
- start=1
- c=11
- c ≠11
Much more important: systems of interacting state-machines typically have many unreachable states!

Example*: a main state-machine (left) controls a second state-machine (a counter).

*)This example was inspired by the dissertation of M.D. Nguyen
We observe that the complete state-space consists of $3 \times 4 = 12$ states, where only 6 states are reachable:
Example of application: prove the **sequential** equivalence of two systems

- The outputs have to be identical for all input sequences
- The outputs have to be identical for all reachable states of the product machine
5.1 Set-Operations with OBDD's

- OBDD-algorithms are also applicable in non-Boolean domains
- Example: set-operations
5. Symbolic Traversal Techniques

5.1 Set-operations with OBDD’s

- **Set-operations:**
  - Starting point: representation of sets by means of characteristic functions
  - Let $S$ be a set with subset $A$, $A \subseteq S$
  - The subset $A$ can be represented by the characteristic function $\chi_A$ as follows:

$$\chi_A(x) = \begin{cases} 
1 & \text{if } x \in A \\
0 & \text{if } x \notin A 
\end{cases}$$
Set-operations can be performed by means of the characteristic functions as follows:

\[ \chi_A \cdot \chi_B \equiv A \cap B, \quad \chi_A + \chi_B \equiv A \cup B, \quad \overline{\chi_A} \equiv \overline{A} \]

Sets are typically encoded by means of Boolean variables and represented by OBDD's, e.g.

<table>
<thead>
<tr>
<th>Element</th>
<th>(x_1)</th>
<th>(x_0)</th>
<th>(\chi_A)</th>
<th>(\chi_B)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A)</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(B)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

\[ \chi_A \cdot \chi_B = (x_1 + \overline{x_0}) \cdot (x_1 + x_0) = x_1 \]
The **Cardinality** (number of elements) of a set $A$, $|A|$, ($\#A$) is calculated as follows:

- Assume $n$ variables
- The leaf-node 1 represents $2^n$ elements
- **Bottom-up procedure:**
  the number of elements $|k|$ of the set characterized by the boolean function of node $k$ is: $1/2^*(|low(k)| + |high(k)|)$

- **Example:**
  $1/2^*(0 + 4) = 2$
  $1/2^*(2 + 4) = 3$
  $1/2^*(0 + 8) = 4$
  $2^3 = 8$
5. Symbolic Traversal Techniques
5.1 Set-operations with OBDD's

Explanation:
- The number of one's is determined by all paths from root-node to leaf-node 1
- The paths correspond to product-terms of a DNF
- Each literal of a product-term divides the number of one's by two (ac \(\sim 2\), abc \(\sim 1\))

\[
\frac{1}{2} \times (|\text{low}(k)| + |\text{high}(k)|)
\]

- \(\frac{1}{2} \times (0 + 4) = 2\)
- \(\frac{1}{2} \times (2 + 4) = 3\)
- \(\frac{1}{2} \times (0 + 8) = 4\)
- \(2^3 = 8\)
Sets with many elements can often be represented by small OBDD's.

Example: the representation of the 393,216 reachable states of the sequential circuit above with 20 flipflops (1,048,576 states) requires an OBDD with 9 nodes.
5. Symbolic Traversal Techniques

5.1 Set-operations with OBDD's

- Representation of binary relations by characteristic functions

Example: \( R \subseteq Q_1 \times Q_2 \)

\[ Q_1 = \{1,3,6\}, \quad Q_2 = \{2,5,6,7\}, \]

\[ R = \{(q_1,q_2) \in Q_1 \times Q_2 \mid q_1 \not< q_2\} = \{(1,2),(1,5),(1,6),(1,7),(3,5),(3,6),(3,7),(6,7)\} \]

Binary encoding:

\[ \begin{align*}
&Q_1: \quad 1 \rightarrow 00 \quad 3 \rightarrow 01 \quad 6 \rightarrow 10 \\
&Q_2: \quad 2 \rightarrow 00 \quad 5 \rightarrow 01 \quad 6 \rightarrow 10 \quad 7 \rightarrow 11
\end{align*} \]

Characteristic function:

\[ \chi_R = \overline{x}_1 \overline{x}_2 + \overline{x}_1 x_2 (y_1 + y_2) + x_1 \overline{x}_2 y_1 y_2 \]
5. Symbolic Traversal Techniques

5.1 Set-operations with OBDD's

Application: determine

\[ K = \{ q_2 \in Q_2 \mid \exists q_1 \in Q_1 : (q_1, q_2) \in R \land (q_1 = 3) \} \]

Compute:

\[ \chi_K = (\exists x_1, x_2 : \chi_R \cdot \bar{x}_1 x_2) = \]
\[ (\exists x_1, x_2 : (\bar{x}_1 \bar{x}_2 + \bar{x}_1 x_2 (y_1 + y_2) + x_1 \bar{x}_2 y_1 y_2) \cdot \bar{x}_1 x_2) = y_1 + y_2 \]
5.2 Representation of Transition Systems

- Many different names for +/- the same objects:
  - Sequential circuits
  - Transition systems
  - Automata
  - Finite-state machines (fsm's)
  - ...

- And many different representations:
  - State-diagrams
  - State-tables
  - Transition-functions
  - ...

5. Symbolic Traversal Techniques

5.2 Representation of transition systems

- State-diagram

Initial state

Notation: input/output
5. Symbolic Traversal Techniques

5.2 Representation of transition systems

- **Definition**: **Mealy automaton** \((S, I, O, \delta, \lambda, s_0)\)
  - Set of states \(S\)
  - Set of input symbols \(I\)
  - Set of output symbols \(O\)
  - Transition function \(\delta: S \times I \rightarrow S\)
  - Output function \(\lambda: S \times I \rightarrow O\)
  - Initial state \(s_0\)

- **Classical definition**

- **Deterministic automaton**
Discrete-time relationships of state-diagrams:

- State at $t+1$ is uniquely determined by state and input at $t$ (transitional behavior)

```
Input at t  Output at t
  0/1
```

```
S0  S1
```

State at $t$  State at $t+1$
State-diagram with **binary encoded states**:
5. Symbolic Traversal Techniques

5.2 Representation of transition systems

- **State-table**

```
<table>
<thead>
<tr>
<th>s_1</th>
<th>s_2</th>
<th>i_1</th>
<th>δ_1</th>
<th>δ_2</th>
<th>λ_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
```

![Diagram of state machine with state transitions labeled 0/0, 1/0, 0/1, 1/1, and 0/0, 1/0, 0/1, 1/1, respectively, and input/output symbol i/o.]
5. Symbolic Traversal Techniques

5.2 Representation of transition systems

- **Transition-functions** \( \delta \) and **output-functions** \( \lambda \)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>( \delta_1 )</th>
<th>( \delta_2 )</th>
<th>( \lambda_1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\[
\delta_1 = s_1 s_2 + s_1 i_1 + \overline{s_1} \overline{s_2} i_1, \\
\delta_2 = s_1 + \overline{i_1}, \\
\lambda_1 = s_1 \overline{s_2} \overline{i_1}
\]
5. Symbolic Traversal Techniques
5.2 Representation of transition systems

- **Structure of a Mealy-Automaton**

![Diagram of a Mealy-Automaton]

- Gate-network
- Storage-elements (flipflops)
- Clock

**Symbols:**
- $i_1, \ldots, i_m$ (inputs)
- $\delta_1, \ldots, \delta_n$ (transition functions)
- $s_1, \ldots, s_n$ (storage elements)
- $\lambda_1, \ldots, \lambda_k$ (outputs)
5. Symbolic Traversal Techniques

5.2 Representation of transition systems

- Structure of a Moore-Automaton
Generally: A Mealy-type transition system with \( n \) storage elements \( s_1, \ldots, s_n \) and \( m \) inputs \( i_1, \ldots, i_m \) is defined by \( n \) transition-functions \( \delta_1, \ldots, \delta_n \)

\[
\delta_i(s_1, \ldots, s_n, i_1, \ldots, i_m)
\]

and \( p \) output-functions \( \lambda_1, \ldots, \lambda_p \) for \( p \) outputs

\[
\lambda_j(s_1, \ldots, s_n, i_1, \ldots, i_m)
\]
Transition-relation

- Introduce "new" state variables $s_i'$
- The transition relation is the characteristic function of the gate-network implementing the transition functions:

$$
T = \prod_{r=1}^{n} (s_r' \equiv \delta_r(s_1, \ldots, s_n, i_1, \ldots, i_m))
$$
5. Symbolic Traversal Techniques

5.2 Representation of transition systems

<table>
<thead>
<tr>
<th>$s_1s_2i$</th>
<th>$s_1's_2'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>00 0</td>
<td>01</td>
</tr>
<tr>
<td>00 1</td>
<td>10</td>
</tr>
<tr>
<td>01 0</td>
<td>01</td>
</tr>
<tr>
<td>01 1</td>
<td>00</td>
</tr>
<tr>
<td>10 0</td>
<td>01</td>
</tr>
<tr>
<td>10 1</td>
<td>11</td>
</tr>
<tr>
<td>11 0</td>
<td>11</td>
</tr>
<tr>
<td>11 1</td>
<td>11</td>
</tr>
</tbody>
</table>

Transition-functions (network of gates)

Outputs define the next state

$s_1' \delta_1 \delta_2$

$s_2' \delta_2$
5. Symbolic Traversal Techniques

5.2 Representation of transition systems

- Example:

\[ T = \prod_{r=1}^{n} (s'_r \equiv \delta_r(s_1, ..., s_n, i_1, ..., i_m)) \]

\[ = (s'_1 \equiv \delta_1(s_1, s_2, i)) \ast (s'_2 \equiv \delta_2(s_1, s_2, i)) \]

The transition relation is 0 for non-existing transitions.
5. Symbolic Traversal Techniques
5.2 Representation of transition systems

- The transition-relation can be obtained easily from the structural representation of a sequential circuit

  Example:

  \[
  T = (s_1' \equiv i) * (s_2' \equiv \overline{s_1}) * (s_3' \equiv s_1)
  \]

  A state-diagram representation is not necessary and may even become infeasible, e.g., for the digital filter
Objective: Determine all states of a transition system which are reachable from the initial state.
Symbolic traversal techniques for transition systems on OBDD-basis

- Coudert et al. (BULL, CAV´89)
- McMillan et al. (CMU, DAC´90)

Two Principles:

- Represent sets of states by means of OBDD´s
- Breadth-first traversal from sets of states to sets of states
**Principle:** Represent sets of states by means of Boolean functions (characteristic functions)

— Example: \( s_1 + s_2 \) characterizes the set \( \{01, 10, 11\} \) of states
5. Symbolic Traversal Techniques

5.3 Symbolic state-space traversal

- Small OBDD's may represent large sets
  - Example: 393,216 of $2^{20}$ states of the filter are reachable, this set is represented by an OBDD of 9 nodes
Principle: Determine all successors of a set of states in breadth-first traversal
Principle: Determine all successors of a set of states in breadth-first traversal
Principle: Determine all successors of a set of states in breadth-first traversal
5. Symbolic Traversal Techniques

5.3 Symbolic state-space traversal

- Depth-first traversal in comparison:
5. Symbolic Traversal Techniques

5.3 Symbolic state-space traversal

- Breadth-first traversal

Diagram:

```
  00
   0 1
  01 10
  01 00 01 11
  11 11
```

```
  01
  0 1
  01 10
  01
  11
```

```
  00
  0 1
  10
```

```
  00
  0 1
  10
```

```
  00
  0 1
```

```
  00
  0 1
```

```
  00
  0 1
```

```
  00
  0 1
```

```
  00
  0 1
```

```
  00
  0 1
```

```
  00
  0 1
```

```
  00
  0 1
```

```
  00
  0 1
```

```
  00
  0 1
```

```
  00
  0 1
```

```
  00
  0 1
```
5. Symbolic Traversal Techniques

5.2 Representation of transition systems

\[
(s_1 \oplus s_2)^+ \quad \text{Bag of reached states}
\]

\[
\begin{align*}
\overline{s_1} \cdot \overline{s_2} \\
01, 10 \\
00 \\
\overline{s_1} \cdot \overline{s_2}
\end{align*}
\]
5. Symbolic Traversal Techniques

5.2 Representation of transition systems

\[ (s_1 \oplus s_2)^+ \]

Bag of reached states

\[ s_1 \cdot \overline{s_2} \]

\[ \overline{s_1} \cdot s_2 \]
5. Symbolic Traversal Techniques

5.2 Representation of transition systems

Bag of reached states

$01 \quad 11$  \quad $s_2^+$

$01 \quad 10$  \quad $(s_1 \oplus s_2)^+$

$00$  \quad $\overline{s_1} \cdot \overline{s_2}$

$s_1 \oplus s_2$

$\overline{s_1} \cdot \overline{s_2}$
5. Symbolic Traversal Techniques

5.2 Representation of transition systems

\[ \text{Bag of reached states} \]

\[
\begin{align*}
01 & \quad 11 & \quad 00 & \quad s_1 + s_2^+ \\
01 & \quad 10 & \quad (s_1 \oplus s_2)^+ \\
00 & \quad & \quad \overline{s_1 \cdot s_2}
\end{align*}
\]
5. Symbolic Traversal Techniques

5.2 Representation of transition systems

\[
\begin{align*}
\text{States at step } n & \\
2 & \overline{s_1 + s_2} \\
1 & \left(s_1 \oplus s_2\right) \\
0 & \overline{s_1 \cdot s_2}
\end{align*}
\]
5. Symbolic Traversal Techniques

5.3 Symbolic state-space traversal

- One step of breadth-first traversal:

![Diagram of transition-function with input and storage elements]
One step of breadth-first traversal:

\[ \chi_{\text{new}}(s_1', s_2') = \exists s_1, s_2, i_1: [s_1' \equiv \delta_1(s_1, s_2, i_1) * s_2' \equiv \delta_2(s_1, s_2, i_1)] * \chi_{\text{old}}(s_1, s_2) \]
5. Symbolic Traversal Techniques

5.3 Symbolic state-space traversal

Transition-relation $T$

$$\chi_{\text{new}}(s_1', s_2') = \exists s_1, s_2, i_1: [(s_1' \equiv \delta_1(s_1, s_2, i_1))*(s_2' \equiv \delta_2(s_1, s_2, i_1))]* \chi_{\text{old}}(s_1, s_2)$$
5. Symbolic Traversal Techniques

5.3 Symbolic state-space traversal

Example:
\[ \exists s_1, s_2, i_1 : (s_1' \equiv \delta_1(s_1, s_2, i_1)) \cdot (s_2' \equiv \delta_2(s_1, s_2, i_1)) \cdot \overline{s}_2 = \]
\[ \exists s_1, s_2, i_1 : (s_1' \equiv s_1 s_2 + s_1 i_1 + \overline{s}_1 \overline{s} \overline{s}_2 i_1) \cdot (s_2' \equiv s_1 + \overline{i}_1) \cdot \overline{s}_2 = \]
\[ \exists s_1, i_1 : (s_1' \equiv i_1) \cdot (s_2' \equiv s_1 + \overline{i}_1) = \]
\[ \exists s_1 : (s_1' \equiv 0) \cdot (s_2' \equiv 1) + (s_1' \equiv 1) \cdot (s_2' \equiv s_1) = \]
\[ \overline{s}_1' s_2' + s_1' \overline{s}_2' + s_1' s_2' \]

\[ \delta_1 = s_1 s_2 + s_1 i_1 + \overline{s}_1 \overline{s} \overline{s}_2 i_1, \]
\[ \delta_2 = s_1 + \overline{i}_1 \]
5. Symbolic Traversal Techniques

5.3 Symbolic state-space traversal

- We have generally:

\[ \chi_{\text{new}}(s'_1, ..., s'_n) = \]

\[ \exists s_1, ..., s_n, i_1, ..., i_m : \prod_{r=1}^{n} (s'_r \equiv \delta_r(s_1, ..., s_n, i_1, ..., i_m))^{*} \chi_{\text{old}}(s_1, ..., s_n) \]

- And also, traversing backwards,:

\[ \chi_{\text{old}}(s_1, ..., s_n) = \]

\[ \exists s'_1, ..., s'_n, i_1, ..., i_m : \prod_{r=1}^{n} (s'_r \equiv \delta_r(s_1, ..., s_n, i_1, ..., i_m))^{*} \chi_{\text{new}}(s'_1, ..., s'_n) \]

- "Maxwell equations" for transition systems
- All operations are Boolean operations and can be implemented by means of OBDD´s
Determine all reachable states of a transition system

- T: transition relation, s0: initial state
- Let \( \text{next}(T, \chi) \) calculate the characteristic function of all successor states characterized by \( \chi \)
- Fixed-point iteration to calculate reached:

\[
\text{traverse}(T, s0):
\begin{align*}
\text{reached} & := s0; \\
\text{from} & := s0; \\
\text{repeat} & \\
\text{to} & := \text{next}(T, \text{from}); \\
\text{new} & := \text{to} \setminus \text{reached}; \\
\text{from} & := \text{new}; \\
\text{reached} & := \text{reached} \cup \text{new}; \\
\text{until} & \text{new} = \emptyset;
\end{align*}
\]
5. Symbolic Traversal Techniques

5.3 Symbolic state-space traversal

traverse(T,s0):
  reached := s0;
  from := s0;
  repeat
    to := next(T, from);
    new := to \ reached;
    from := new;
    reached := reached \cup new;
  until new = \emptyset;

Set-operations:

traverse(T,s0):
  reached := s0;
  from := s0;
  repeat
    to := next(T, from);
    new := to \setminus reached;
    from := new;
    reached := reached \cup new;
  until new = \emptyset;

OBDD's:
"from" can be chosen arbitrarily if
\[ \text{new} \subseteq \text{from} \subseteq \text{reached} \cup \text{new} \]

Since \( \text{next}(T, \text{from}) \) is in terms of the "new" state-variables \( s_i' \), we have to replace the "new" by the "old" state-variables after each step

\[
f(s' \leftarrow s) = s \cdot f_s' + \overline{s} \cdot f_s'.
\]
5. Symbolic Traversal Techniques

5.3 Symbolic state-space traversal

- Breadth-first traversal
Symbolic state-space traversal is applicable to systems where it is not possible to derive the state-diagram, i.e., where only transition-functions are available, e.g.:
What about communicating synchronous machines?

The number of states of the so-called product machine equals the product of the number of states of the individual machines.
The transition relation of the product machine is the boolean product of the transition-relations of the individual machines

\[ \chi_{\text{new}}(s'_1, s'_2, s'_3) = \exists s_1, s_2, s_3, i: (s'_1 \equiv \delta_1(s_1, s_2, s_3, i))^* \]
\[ (s'_2 \equiv \delta_2(s_1, s_2, s_3, i))^* (s'_3 \equiv \delta_3(s_1, s_2, s_3, i))^* \chi_{\text{old}}(s_1, s_2, s_3) \]
5.4 Equivalence of Sequential Circuits

- **Case 1**: The state encoding of two systems is equal
  - The problem can be reduced to logic verification
- **Case 2**: The state encoding is not equal, e.g.:
For case 2, the product machine has to be built

- Both machines are connected with the same inputs
The product machine is viewed as one machine

The reachable states "reached" are computed

Prove that \( \forall I: \text{reached} \rightarrow (O_1 \equiv O_2) \) for all corresponding outputs
5. Symbolic Traversal Techniques

5.4 Equivalence of sequential circuits

Application (BULL, AT&T, ...):

- Specification
- Synthesis
- VHDL-Description
- Synthesis
- Transistor Netlist
- Synthesis
- Layout

Equivalence checking

- FSM 1
- Synthesis (modifies state encoding !)
- Extraction

- FSM 2
- Extraction

Transistor Netlist
5. Symbolic Traversal Techniques

5.5 Efficient Forward Traversal

- "Early quantification" as an example of an efficient forward traversal technique
5. Symbolic Traversal Techniques

5.5 Efficient forward traversal

- A single forward traversal step:
  \[
  \chi_{\text{new}}(s_1', ..., s_n') = \exists s_1, ..., s_n, i_1, ..., i_m:
  \prod (s_r' \equiv \delta_r(s_1, ..., s_n, i_1, ..., i_m))^* \chi_{\text{old}}(s_1, ..., s_n)
  \]

- The mechanization of forward traversal means
  - Build the transition relation
    - A huge monolithic product-term!
  - Build product of transition relation and \( \chi_{\text{old}}(s_1, ..., s_n) \)
  - Then: **existential quantification** of all old state-variables and input-variables in the resulting big product term
    - Not feasible for large circuits
    - Idea: apply existential quantification to each of the products separately???
5. Symbolic Traversal Techniques

5.5 Efficient forward traversal

- Some properties of quantified boolean formulas (QBF’s):

\[(\forall x: f(x) \cdot g(x)) = (\forall x: f(x)) \cdot (\forall x: g(x)),\]
\[(\exists x: f(x) + g(x)) = (\exists x: f(x)) + (\exists x: g(x))\]

Def.: \( f \leq g \) iff \( (f \rightarrow g) = 1 \)

\[(\forall x: f(x)) + (\forall x: g(x)) \leq (\forall x: f(x) + g(x)),\]
\[(\exists x: f(x) \cdot g(x)) \leq (\exists x: f(x)) \cdot (\exists x: g(x))\]

\[= f_x g_x + f_x g_x = (f_x + f_x) \cdot (g_x + g_x)\]
\[= f_x g_x + f_x g_x + f_x g_x + f_x g_x\]
Simplifications are possible if the products depend on disjunct sets of variables.

We have:

\[(\exists x: f(x) \times g(x)) \leq (\exists x: f(x)) \times (\exists x: g(x))\]

but, if f does not depend on x, we have

\[(\exists x: f \times g(x)) = f \times (\exists x: g(x))\]
5. Symbolic Traversal Techniques

5.5 Efficient forward traversal

— Example:

\[ \exists s_1, s_2, i_1 : (s'_1 \equiv s_1 s_2 + s_1 i_1 + \overline{s_1} \overline{s_2} i_1) \cdot (s'_2 \equiv s_1 + \overline{i_1}) \cdot \overline{s_1} = \]

\[ \exists s_1, i_1 : \left[ \exists s_2 : (s'_1 \equiv s_1 s_2 + s_1 i_1 + \overline{s_1} \overline{s_2} i_1) \right] \cdot (s'_2 \equiv s_1 + \overline{i_1}) \cdot \overline{s_1} \]
If the transition-relation $T = T_1 \ast \ldots \ast T_n$ can be partitioned into $r$ clusters $C_i$, $T = C_1 \ast \ldots \ast C_r$, where for the supporting variable-sets $S_i$ of the clusters $C_i$ holds:

$$\exists S_1: (C_1 * (\exists S_2: (C_2 * \ldots * (\exists S_r: (C_r * \chi_0)) \ldots ))))$$

$C_1, \ldots, C_{r-1}$ does not depend on $S_r$

$C_1, \ldots, C_{r-2}$ do not depend on $S_{r-1}$

etc.

then

$$\exists S: T^*\chi_0 = (\exists S_1: (C_1 * (\exists S_2: (C_2 * \ldots * (\exists S_r: (C_r * \chi_0)) \ldots ))))$$
Clustered dependencies are typical for interacting transition systems, e.g.,
5. Symbolic Traversal Techniques

5.5 Efficient forward traversal

- Idea + benefits:

  \[ f(x_1, \ldots, x_k, \ldots, x_n) \ast g(x_1, \ldots, x_k) : \]
  \[ \text{recursion stops at } k ! \]

  \[ f(x_1, \ldots, x_k, \ldots, x_n) \ast h(x_k, \ldots, x_n) : \]
  \[ f \text{ from } x_1 \text{ to } x_{k-1} \text{ is unchanged!} \]

\[
(\exists x_1, x_2 : (\delta_1 \ast (\exists x_3 : (\delta_2 \ast \delta_3 \ast (\exists x_4, x_5 : \delta_4 \ast \delta_5 \ast \delta_6 ))))))
\]