Semantics and Verification of Software

Lecture 20: Dataflow Analysis

Thomas Noll

Lehrstuhl für Informatik 2
RWTH Aachen University
noll@cs.rwth-aachen.de

http://www-i2.informatik.rwth-aachen.de/i2/svsw/

Summer semester 2007

Outline

1. Repetition: Fixpoint and MOP Solution
2. MOP vs. Fixpoint Solution
3. Diplomarbeit/Master Thesis
4. Evaluation of the Course

Fixpoint Solution I

Just as in the denotational semantics of `while` loops, the equation system determines a functional whose fixpoints are the solutions of the equation system.

Definition (Dataflow functional)

The equation system of a dataflow system $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$ induces a functional

$$\Phi_S : D^n \to D^n : (d_{l_1}, \ldots, d_{l_n}) \mapsto (d'_{l_1}, \ldots, d'_{l_n})$$

where $Lab = \{l_1, \ldots, l_n\}$ and

$$d'_{l_i} := \begin{cases} \iota & \text{if } l \in E \\ \bigsqcup\{\varphi_{l'}(d_{l'}) \mid (l', l_i) \in F\} & \text{otherwise} \end{cases}$$

Fixpoint Solution II

Remarks:

- $(d_1, \ldots, d_n)$ is a solution of the equation system iff it is a fixpoint of Φ_S
- If $(D, \sqsubseteq)$ is a complete lattice satisfying ACC, then so is $(D^n, \sqsubseteq^n)$ (where $(d_1, \ldots, d_n) \sqsubseteq^n (d'_1, \ldots, d'_n)$ iff $d_i \sqsubseteq d'_i$ for every $1 \leq i \leq n$)
- Every transfer function φ_l monotonic in D
  $\implies$ Φ_S monotonic in D^n
- Thus the fixpoint is effectively computable by iteration:

$$\mathsf{fix}(\Phi_S) = \bigsqcup \{\Phi_S^i(\bot_{D^n}) \mid i \in \mathbb{N}\}$$

  where $\bot_{D^n} = (\underbrace{\bot_D, \ldots, \bot_D}_{n \text{ times}})$
- If maximal length of chains in D is m
  $\implies$ maximal length of chains in D^n is $m \cdot n$
  $\implies$ fixpoint iteration requires at most $m \cdot n$ steps

MOP Solution I

- Other solution method for dataflow systems
- MOP = Meet Over all Paths
- Analysis information for block B^l := least upper bound over all paths leading to l

Definition (Paths)

Let $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$ be a dataflow system. For every $l \in Lab$, the set of paths up to l is given by

$$Path(l) := \{[l_1, \ldots, l_{k-1}] \mid k \geq 1, l_1 \in E, (l_i, l_{i+1}) \in F \text{ for every } 1 \leq i \leq k, l_k = l\}.$$

For a path $p = [l_1, \ldots, l_{k-1}] \in Path(l)$, we define the transfer function $\varphi_p : D \to D$ by

$$\varphi_p := \varphi_{l_{k-1}} \circ \ldots \circ \varphi_{l_1} \circ \mathsf{id}_D$$

(so that $\varphi_{[]} = \mathsf{id}_D$).

MOP Solution II

Definition (MOP solution)

Let $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$ be a dataflow system where $Lab = \{l_1, \ldots, l_n\}$. The MOP solution for S is determined by

$$\mathsf{mop}(S) := (\mathsf{mop}(l_1), \ldots, \mathsf{mop}(l_n)) \in D^n$$

where, for every $l \in Lab$,

$$\mathsf{mop}(l) := \bigsqcup\{\varphi_p(\iota) \mid p \in Path(l)\}.$$

Remark:

- $Path(l)$ is generally infinite

$\Longrightarrow$ not clear how to compute $\mathsf{mop}(l)$

- In fact: MOP solution generally undecidable (later)

Undecidability of MOP Solution

Theorem (Undecidability of MOP solution)

The MOP solution for Constant Propagation is undecidable.

Proof.

Based on undecidability of Modified Post Correspondence Problem:
Let Γ be some alphabet, $n \in \mathbb{N}$, and $u_1, \ldots, u_n, v_1, \ldots, v_n \in \Gamma^+$.
Does there exist $i_1, \ldots, i_m \in \{1, \ldots, n\}$ with $m \geq 1$ and $i_1 = 1$ such that $u_{i_1} u_{i_2} \ldots u_{i_m} = v_{i_1} v_{i_2} \ldots v_{i_m}$?

(on the board) □

Outline

MOP vs. Fixpoint Solution I

Theorem 20.1 (MOP vs. Fixpoint Solution)

Let $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$ *be a dataflow system. Then*

$$\mathsf{mop}(S) \sqsubseteq \mathsf{fix}(\Phi_S)$$

Proof.

on the board □

The next example shows that both solutions can indeed be different.

MOP vs. Fixpoint Solution I

Theorem 20.1 (MOP vs. Fixpoint Solution)

Let $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$ *be a dataflow system. Then*

$$\mathsf{mop}(S) \sqsubseteq \mathsf{fix}(\Phi_S)$$

Proof.

on the board □

The next example shows that both solutions can indeed be different.

MOP vs. Fixpoint Solution I

Theorem 20.1 (MOP vs. Fixpoint Solution)

Let $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$ *be a dataflow system. Then*

$$\mathsf{mop}(S) \sqsubseteq \mathsf{fix}(\Phi_S)$$

Proof.

on the board □

The next example shows that both solutions can indeed be different.

MOP vs. Fixpoint Solution II

Example 20.2 (Constant Propagation)

```
c := if [z > 0]^1 then
       [x := 2;]^2
       [y := 3;]^3
     else
       [x := 3;]^4
       [y := 2;]^5
     [z := x+y;]^6
     [...]^7
```

Transfer functions (for $\delta = (\delta(\mathrm{x}), \delta(\mathrm{y}), \delta(\mathrm{z})) \in D$):

$\varphi_1((a, b, c)) = (a, b, c)$
$\varphi_2((a, b, c)) = (2, b, c)$
$\varphi_3((a, b, c)) = (a, 3, c)$
$\varphi_4((a, b, c)) = (3, b, c)$
$\varphi_5((a, b, c)) = (a, 2, c)$
$\varphi_6((a, b, c)) = (a, b, a + b)$

1. Fixpoint solution:

$$\begin{aligned}
\mathsf{CP}_1 &= \iota &&= (\top, \top, \top)\\
\mathsf{CP}_2 &= \varphi_1(\mathsf{CP}_1) &&= (\top, \top, \top)\\
\mathsf{CP}_3 &= \varphi_2(\mathsf{CP}_2) &&= (2, \top, \top)\\
\mathsf{CP}_4 &= \varphi_1(\mathsf{CP}_1) &&= (\top, \top, \top)\\
\mathsf{CP}_5 &= \varphi_2(\mathsf{CP}_2) &&= (3, \top, \top)\\
\mathsf{CP}_6 &= \varphi_3(\mathsf{CP}_3) \sqcup \varphi_5(\mathsf{CP}_5)\\
&= (2, 3, \top) \sqcup (3, 2, \top) &&= (\top, \top, \top)\\
\mathsf{CP}_7 &= \varphi_6(\mathsf{CP}_6) &&= (\top, \top, \top)
\end{aligned}$$

2. MOP solution:

$$\begin{aligned}
\mathsf{mop}(7) &= \varphi_{[1,2,3,6]}(\top, \top, \top) \sqcup \varphi_{[1,4,5,6]}(\top, \top, \top)\\
&= (2, 3, 5) \sqcup (3, 2, 5)\\
&= (\top, \top, 5)
\end{aligned}$$

MOP vs. Fixpoint Solution II

Example 20.2 (Constant Propagation)

```
c := if [z > 0]^1 then
        [x := 2;]^2
        [y := 3;]^3
     else
        [x := 3;]^4
        [y := 2;]^5
     [z := x+y;]^6
     [...]^7
```

Transfer functions (for $\delta = (\delta(\mathtt{x}), \delta(\mathtt{y}), \delta(\mathtt{z})) \in D$):

$$\varphi_1((a,b,c)) = (a,b,c)$$
$$\varphi_2((a,b,c)) = (2,b,c)$$
$$\varphi_3((a,b,c)) = (a,3,c)$$
$$\varphi_4((a,b,c)) = (3,b,c)$$
$$\varphi_5((a,b,c)) = (a,2,c)$$
$$\varphi_6((a,b,c)) = (a,b,a+b)$$

1. Fixpoint solution:

$$\begin{aligned}
\mathsf{CP}_1 &= \iota && = (\top,\top,\top)\\
\mathsf{CP}_2 &= \varphi_1(\mathsf{CP}_1) && = (\top,\top,\top)\\
\mathsf{CP}_3 &= \varphi_2(\mathsf{CP}_2) && = (2,\top,\top)\\
\mathsf{CP}_4 &= \varphi_1(\mathsf{CP}_1) && = (\top,\top,\top)\\
\mathsf{CP}_5 &= \varphi_2(\mathsf{CP}_2) && = (3,\top,\top)\\
\mathsf{CP}_6 &= \varphi_3(\mathsf{CP}_3) \sqcup \varphi_5(\mathsf{CP}_5) &&\\
&= (2,3,\top) \sqcup (3,2,\top) && = (\top,\top,\top)\\
\mathsf{CP}_7 &= \varphi_6(\mathsf{CP}_6) && = (\top,\top,\top)
\end{aligned}$$

2. MOP solution:

$$\begin{aligned}
\mathsf{mop}(7) &= \varphi_{[1,2,3,6]}(\top,\top,\top) \sqcup \\
&\quad\ \varphi_{[1,4,5,6]}(\top,\top,\top)\\
&= (2,3,5) \sqcup (3,2,5)\\
&= (\top,\top,5)
\end{aligned}$$

MOP vs. Fixpoint Solution II

Example 20.2 (Constant Propagation)

```
c := if [z > 0]^1 then
       [x := 2;]^2
       [y := 3;]^3
     else
       [x := 3;]^4
       [y := 2;]^5
     [z := x+y;]^6
     [...]^7
```

Transfer functions (for $\delta = (\delta(\mathtt{x}), \delta(\mathtt{y}), \delta(\mathtt{z})) \in D$):

$$\begin{aligned}
\varphi_1((a,b,c)) &= (a,b,c)\\
\varphi_2((a,b,c)) &= (2,b,c)\\
\varphi_3((a,b,c)) &= (a,3,c)\\
\varphi_4((a,b,c)) &= (3,b,c)\\
\varphi_5((a,b,c)) &= (a,2,c)\\
\varphi_6((a,b,c)) &= (a,b,a+b)
\end{aligned}$$

1. Fixpoint solution:

$$\begin{aligned}
\mathsf{CP}_1 &= \iota && = (\top,\top,\top)\\
\mathsf{CP}_2 &= \varphi_1(\mathsf{CP}_1) && = (\top,\top,\top)\\
\mathsf{CP}_3 &= \varphi_2(\mathsf{CP}_2) && = (2,\top,\top)\\
\mathsf{CP}_4 &= \varphi_1(\mathsf{CP}_1) && = (\top,\top,\top)\\
\mathsf{CP}_5 &= \varphi_2(\mathsf{CP}_2) && = (3,\top,\top)\\
\mathsf{CP}_6 &= \varphi_3(\mathsf{CP}_3) \sqcup \varphi_5(\mathsf{CP}_5) &&\\
&= (2,3,\top) \sqcup (3,2,\top) && = (\top,\top,\top)\\
\mathsf{CP}_7 &= \varphi_6(\mathsf{CP}_6) && = (\top,\top,\top)
\end{aligned}$$

2. MOP solution:

$$\begin{aligned}
\mathsf{mop}(7) &= \varphi_{[1,2,3,6]}(\top,\top,\top) \sqcup\\
&\quad\ \varphi_{[1,4,5,6]}(\top,\top,\top)\\
&= (2,3,5) \sqcup (3,2,5)\\
&= (\top,\top,5)
\end{aligned}$$

MOP vs. Fixpoint Solution II

Example 20.2 (Constant Propagation)

c := `if [z > 0]`^{1} `then`

```
      [x := 2;]²
      [y := 3;]³
    else
      [x := 3;]⁴
      [y := 2;]⁵
    [z := x+y;]⁶
    [...]⁷
```

Transfer functions (for $\delta = (\delta(\mathtt{x}), \delta(\mathtt{y}), \delta(\mathtt{z})) \in D$):

$\varphi_1((a,b,c)) = (a,b,c)$
$\varphi_2((a,b,c)) = (2,b,c)$
$\varphi_3((a,b,c)) = (a,3,c)$
$\varphi_4((a,b,c)) = (3,b,c)$
$\varphi_5((a,b,c)) = (a,2,c)$
$\varphi_6((a,b,c)) = (a,b,a+b)$

1. Fixpoint solution:

$$\begin{aligned}
\mathsf{CP}_1 &= \iota && = (\top,\top,\top)\\
\mathsf{CP}_2 &= \varphi_1(\mathsf{CP}_1) && = (\top,\top,\top)\\
\mathsf{CP}_3 &= \varphi_2(\mathsf{CP}_2) && = (2,\top,\top)\\
\mathsf{CP}_4 &= \varphi_1(\mathsf{CP}_1) && = (\top,\top,\top)\\
\mathsf{CP}_5 &= \varphi_2(\mathsf{CP}_2) && = (3,\top,\top)\\
\mathsf{CP}_6 &= \varphi_3(\mathsf{CP}_3) \sqcup \varphi_5(\mathsf{CP}_5) &&\\
&= (2,3,\top) \sqcup (3,2,\top) && = (\top,\top,\top)\\
\mathsf{CP}_7 &= \varphi_6(\mathsf{CP}_6) && = (\top,\top,\top)
\end{aligned}$$

2. MOP solution:

$$\begin{aligned}
\mathsf{mop}(7) &= \varphi_{[1,2,3,6]}(\top,\top,\top) \sqcup \\
&\quad\ \varphi_{[1,4,5,6]}(\top,\top,\top)\\
&= (2,3,5) \sqcup (3,2,5)\\
&= (\top,\top,5)
\end{aligned}$$

Distributive Transfer Functions I

A sufficient criterion for the coincidenece of MOP and Fixpoint Solution is the distributivity of the transfer functions.

Definition 20.3 (Distributivity)

- Let $(D, \sqsubseteq)$ and $(D', \sqsubseteq')$ be complete lattices, and let $F : D \to D'$. F is called distributive (w.r.t. $(D, \sqsubseteq)$ and $(D', \sqsubseteq')$) if, for every $d_1, d_2 \in D$,
$$F(d_1 \sqcup_D d_2) = F(d_1) \sqcup_{D'} F(d_2).$$
- A dataflow system $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$ is called distributive if every $\varphi_l : D \to D$ is so.

Distributive Transfer Functions I

A sufficient criterion for the coincidenece of MOP and Fixpoint Solution is the distributivity of the transfer functions.

Definition 20.3 (Distributivity)

- Let $(D, \sqsubseteq)$ and $(D', \sqsubseteq')$ be complete lattices, and let $F : D \to D'$. F is called distributive (w.r.t. $(D, \sqsubseteq)$ and $(D', \sqsubseteq')$) if, for every $d_1, d_2 \in D$,
$$F(d_1 \sqcup_D d_2) = F(d_1) \sqcup_{D'} F(d_2).$$
- A dataflow system $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$ is called distributive if every $\varphi_l : D \to D$ is so.

Example 20.4

1. The Available Expressions dataflow system is distributive:

$$\begin{aligned}\varphi_l(A_1 \sqcup A_2) &= ((A_1 \cap A_2) \setminus \mathsf{kill}_{\mathsf{AE}}(B^l)) \cup \mathsf{gen}_{\mathsf{AE}}(B^l) \\ &= ((A_1 \setminus \mathsf{kill}_{\mathsf{AE}}(B^l)) \cup \mathsf{gen}_{\mathsf{AE}}(B^l)) \cap \\ &\quad ((A_2 \setminus \mathsf{kill}_{\mathsf{AE}}(B^l)) \cup \mathsf{gen}_{\mathsf{AE}}(B^l)) \\ &= \varphi_l(A_1) \sqcup \varphi_l(A_2)\end{aligned}$$

2. The Live Variables dataflow system is distributive (similar)
3. The Constant Propagation dataflow system is not distributive:

$$\begin{aligned}(\top,\top,\top) &= \varphi_{\texttt{z:=x+y}}((2,3,\top) \sqcup (3,2,\top)) \\ &\neq \varphi_{\texttt{z:=x+y}}((2,3,\top)) \sqcup \varphi_{\texttt{z:=x+y}}((3,2,\top)) \\ &= (\top,\top,5)\end{aligned}$$

Distributive Transfer Functions II

Example 20.4

1. The Available Expressions dataflow system is distributive:

$$\begin{aligned}\varphi_l(A_1 \sqcup A_2) &= ((A_1 \cap A_2) \setminus \mathsf{kill}_{\mathsf{AE}}(B^l)) \cup \mathsf{gen}_{\mathsf{AE}}(B^l) \\ &= ((A_1 \setminus \mathsf{kill}_{\mathsf{AE}}(B^l)) \cup \mathsf{gen}_{\mathsf{AE}}(B^l)) \cap \\ &\quad\; ((A_2 \setminus \mathsf{kill}_{\mathsf{AE}}(B^l)) \cup \mathsf{gen}_{\mathsf{AE}}(B^l)) \\ &= \varphi_l(A_1) \sqcup \varphi_l(A_2)\end{aligned}$$

2. The Live Variables dataflow system is distributive (similar)

3. The Constant Propagation dataflow system is not distributive:

$$\begin{aligned}(\top,\top,\top) &= \varphi_{\mathtt{z:=x+y}}((2,3,\top) \sqcup (3,2,\top)) \\ &\neq \varphi_{\mathtt{z:=x+y}}((2,3,\top)) \sqcup \varphi_{\mathtt{z:=x+y}}((3,2,\top)) \\ &= (\top,\top,5)\end{aligned}$$

Distributive Transfer Functions II

Example 20.4

1. The Available Expressions dataflow system is distributive:

$$\begin{aligned}\varphi_l(A_1 \sqcup A_2) &= ((A_1 \cap A_2) \setminus \mathsf{kill}_{\mathsf{AE}}(B^l)) \cup \mathsf{gen}_{\mathsf{AE}}(B^l) \\ &= ((A_1 \setminus \mathsf{kill}_{\mathsf{AE}}(B^l)) \cup \mathsf{gen}_{\mathsf{AE}}(B^l)) \cap \\ &\quad\ ((A_2 \setminus \mathsf{kill}_{\mathsf{AE}}(B^l)) \cup \mathsf{gen}_{\mathsf{AE}}(B^l)) \\ &= \varphi_l(A_1) \sqcup \varphi_l(A_2)\end{aligned}$$

2. The Live Variables dataflow system is distributive (similar)
3. The Constant Propagation dataflow system is not distributive:

$$\begin{aligned}(\top, \top, \top) &= \varphi_{\texttt{z:=x+y}}((2, 3, \top) \sqcup (3, 2, \top)) \\ &\neq \varphi_{\texttt{z:=x+y}}((2, 3, \top)) \sqcup \varphi_{\texttt{z:=x+y}}((3, 2, \top)) \\ &= (\top, \top, 5)\end{aligned}$$

Coincidence of MOP and Fixpoint Solution

Theorem 20.5 (MOP vs. Fixpoint Solution)

Let $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$ *be a distributive dataflow system. Then*

$$\mathsf{mop}(S) = \mathsf{fix}(\Phi_S)$$

Proof.

- by showing that $\Phi_S(\mathsf{mop}(S)) = \mathsf{mop}(S)$ ...
  (see [Nielson/Nielson/Hankin 2005, p. 81])
- ... and using $\mathsf{mop}(S) \sqsubseteq \mathsf{fix}(\Phi_S)$ (Theorem 20.1)

□

Coincidence of MOP and Fixpoint Solution

Theorem 20.5 (MOP vs. Fixpoint Solution)

Let $S = (Lab, E, F, (D, \sqsubseteq), \iota, \varphi)$ *be a distributive dataflow system. Then*

$$\mathsf{mop}(S) = \mathsf{fix}(\Phi_S)$$

Proof.

- by showing that $\Phi_S(\mathsf{mop}(S)) = \mathsf{mop}(S)$ ...
  (see [Nielson/Nielson/Hankin 2005, p. 81])
- ... and using $\mathsf{mop}(S) \sqsubseteq \mathsf{fix}(\Phi_S)$ (Theorem 20.1)

□

Outline

Model Checking Microcontroller Assembly Code

- Motivation: microcontrollers frequently employed in embedded systems
- Embedded systems often safety–critical (cars, planes, medical devices, ...)
- Exhaustive testing generally impossible (uncertain environments, huge state spaces, ...)

$\Longrightarrow$ Formal reasoning methods

- Here: Model Checking system $\overset{?}{\models}$ specification

  system: (semantics of) assembly code $\Longrightarrow$ labeled transition system

  specification: formula of some temporal logic

  - never two processes in critical section: $AG\ \neg(crit_1 \wedge crit_2)$
  - every request will be answered before timeout: $AG\ (req \implies \neg\ timeout\ U\ answer)$

Model Checking Microcontroller Assembly Code

- Motivation: microcontrollers frequently employed in embedded systems
- Embedded systems often safety–critical (cars, planes, medical devices, ...)
- Exhaustive testing generally impossible (uncertain environments, huge state spaces, ...)

$\Longrightarrow$ Formal reasoning methods

- Here: Model Checking system $\overset{?}{\models}$ specification

  system: (semantics of) assembly code $\Longrightarrow$ labeled transition system

  specification: formula of some temporal logic

  - never two processes in critical section: $AG\ \neg(crit_1 \wedge crit_2)$
  - every request will be answered before timeout: $AG\ (req \implies \neg\ timeout\ U\ answer)$

Model Checking Microcontroller Assembly Code

- Motivation: microcontrollers frequently employed in embedded systems
- Embedded systems often safety–critical (cars, planes, medical devices, ...)
- Exhaustive testing generally impossible (uncertain environments, huge state spaces, ...)

$\Longrightarrow$ Formal reasoning methods

- Here: Model Checking system $\stackrel{?}{\models}$ specification

  system: (semantics of) assembly code $\Longrightarrow$ labeled transition system

  specification: formula of some temporal logic

  - never two processes in critical section: $AG\ \neg(crit_1 \wedge crit_2)$
  - every request will be answered before timeout: $AG\ (req \implies \neg\ timeout\ U\ answer)$

- Motivation: microcontrollers frequently employed in embedded systems
- Embedded systems often safety–critical (cars, planes, medical devices, ...)
- Exhaustive testing generally impossible (uncertain environments, huge state spaces, ...)

$\Longrightarrow$ Formal reasoning methods

- Here: Model Checking system $\overset{?}{\models}$ specification

  system: (semantics of) assembly code $\Longrightarrow$ labeled transition system

  specification: formula of some temporal logic

  - never two processes in critical section: $AG\ \neg(crit_1 \wedge crit_2)$
  - every request will be answered before timeout: $AG\ (req \Longrightarrow \neg\ timeout\ U\ answer)$

Model Checking Microcontroller Assembly Code

- Motivation: microcontrollers frequently employed in embedded systems
- Embedded systems often safety–critical (cars, planes, medical devices, ...)
- Exhaustive testing generally impossible (uncertain environments, huge state spaces, ...)

$\Longrightarrow$ Formal reasoning methods

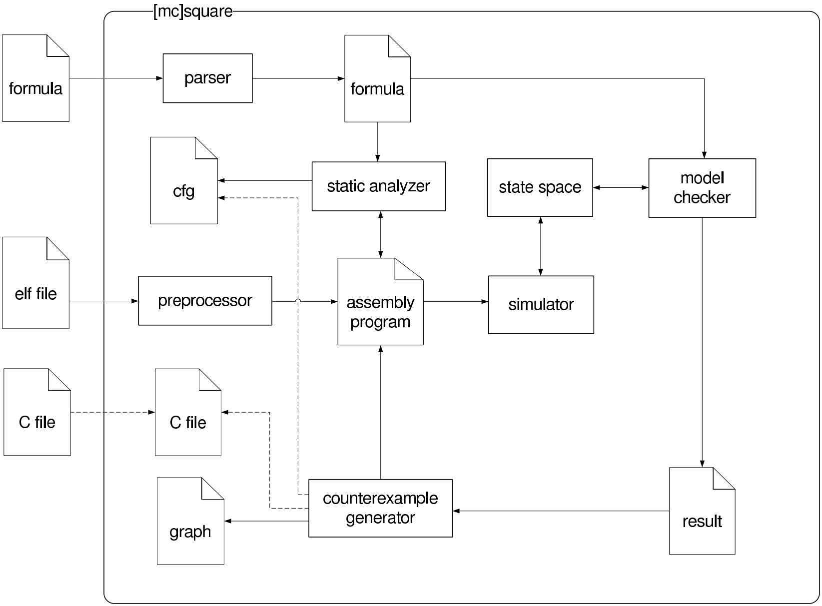
- Here: Model Checking system $\stackrel{?}{\models}$ specification

  system: (semantics of) assembly code $\Longrightarrow$ labeled transition system

  specification: formula of some temporal logic

  - never two processes in critical section: $AG\ \neg(crit_1 \wedge crit_2)$
  - every request will be answered before timeout: $AG\ (req \Longrightarrow \neg\ timeout\ U\ answer)$

The [MC]square Tool



Current State

- Currently supported microprocessors:
  - Atmel ATmega 16
  - Infineon XC167
- State–space generator written by hand for each microprocessor
- Desirable: compiler–generating approach

$$\text{microprocessor specification} \longrightarrow \text{state–space generator}$$

- (Parts of) formal model available:
  - Interrupt handler:

$$\texttt{SREG[I]} = 1 \wedge \texttt{TIMSK[TOIE0]} = 1 \wedge \texttt{TIFR[TOV0]} = 1 \rightarrow : 18 \;>$$
$$\texttt{SREG[I]} = 1 \wedge \texttt{GICR[INT2]} = 1 \wedge \texttt{GIFR[INTF2]} = 1 \rightarrow : 36 > \ldots$$

  - Instruction handler (here: ADD Ri,Rj at address q):

$$q : \texttt{R}i := \texttt{R}i + \texttt{R}j, \texttt{SREG[Z]} := (\texttt{R}i + \texttt{R}j = 0), \texttt{SREG[C]} := \ldots, \ldots : q + 2$$

Current State

- Currently supported microprocessors:
  - Atmel ATmega 16
  - Infineon XC167
- State–space generator written by hand for each microprocessor
- Desirable: compiler–generating approach

$$\text{microprocessor specification} \longrightarrow \text{state–space generator}$$

- (Parts of) formal model available:
  - Interrupt handler:

$$\texttt{SREG[I]} = 1 \wedge \texttt{TIMSK[TOIE0]} = 1 \wedge \texttt{TIFR[TOV0]} = 1 \rightarrow : 18 \; >$$
$$\texttt{SREG[I]} = 1 \wedge \texttt{GICR[INT2]} = 1 \wedge \texttt{GIFR[INTF2]} = 1 \rightarrow : 36 > \ldots$$

  - Instruction handler (here: ADD Ri,Rj at address q):

$$q : \texttt{R}i := \texttt{R}i + \texttt{R}j, \texttt{SREG[Z]} := (\texttt{R}i + \texttt{R}j = 0), \texttt{SREG[C]} := \ldots, \ldots : q + 2$$

- Currently supported microprocessors:
  - Atmel ATmega 16
  - Infineon XC167
- State–space generator written by hand for each microprocessor
- Desirable: compiler–generating approach

$$\text{microprocessor specification} \longrightarrow \text{state–space generator}$$

- (Parts of) formal model available:
  - Interrupt handler:

$$\begin{array}{l}\texttt{SREG[I]} = 1 \wedge \texttt{TIMSK[TOIE0]} = 1 \wedge \texttt{TIFR[TOV0]} = 1 \longrightarrow: 18 \;> \\ \texttt{SREG[I]} = 1 \wedge \texttt{GICR[INT2]} = 1 \wedge \texttt{GIFR[INTF2]} = 1 \longrightarrow: 36 > \ldots\end{array}$$

  - Instruction handler (here: ADD Ri,Rj at address q):

$$q : \texttt{R}i := \texttt{R}i + \texttt{R}j, \texttt{SREG[Z]} := (\texttt{R}i + \texttt{R}j = 0), \texttt{SREG[C]} := \ldots, \ldots : q + 2$$

Current State

- Currently supported microprocessors:
  - Atmel ATmega 16
  - Infineon XC167
- State–space generator written by hand for each microprocessor
- Desirable: compiler–generating approach

$$\text{microprocessor specification} \longrightarrow \text{state–space generator}$$

- (Parts of) formal model available:
  - Interrupt handler:

$$\begin{array}{l}\texttt{SREG[I]} = 1 \land \texttt{TIMSK[TOIE0]} = 1 \land \texttt{TIFR[TOV0]} = 1 \rightarrow: 18\ > \\ \texttt{SREG[I]} = 1 \land \texttt{GICR[INT2]} = 1 \land \texttt{GIFR[INTF2]} = 1 \rightarrow: 36 > \ldots\end{array}$$

  - Instruction handler (here: `ADD R`i`,R`j at address q):

$$q : \texttt{R}i := \texttt{R}i + \texttt{R}j, \texttt{SREG[Z]} := (\texttt{R}i + \texttt{R}j = 0), \texttt{SREG[C]} := \ldots, \ldots : q + 2$$

The Thesis

Goal:

- Tool for automatic generation of (parts of) state–space generator from microprocessor specification
- Embedded in [mc]square environment
- Support of state–space abstraction techniques ("delayed nondeterminism")
- Case study: Motorola ARM 7

Desirable prerequesites:

- Formal Methods for Embedded Systems [Kowalewski]
- Model Checking [Katoen, Thomas]
- Compiler Construction [Indermark, Noll]
- Semantics and Verification of Software [Noll]

Contact:

- Bastian Schlich (Inf. 11, `schlich@cs.rwth-aachen.de`)
- Thomas Noll (Inf. 2, `noll@cs.rwth-aachen.de`)

The Thesis

Goal:

- Tool for automatic generation of (parts of) state–space generator from microprocessor specification
- Embedded in [mc]square environment
- Support of state–space abstraction techniques ("delayed nondeterminism")
- Case study: Motorola ARM 7

Desirable prerequesites:

- Formal Methods for Embedded Systems [Kowalewski]
- Model Checking [Katoen, Thomas]
- Compiler Construction [Indermark, Noll]
- Semantics and Verification of Software [Noll]

Contact:

- Bastian Schlich (Inf. 11, `schlich@cs.rwth-aachen.de`)
- Thomas Noll (Inf. 2, `noll@cs.rwth-aachen.de`)

The Thesis

Goal:

- Tool for automatic generation of (parts of) state–space generator from microprocessor specification
- Embedded in [mc]square environment
- Support of state–space abstraction techniques ("delayed nondeterminism")
- Case study: Motorola ARM 7

Desirable prerequesites:

- Formal Methods for Embedded Systems [Kowalewski]
- Model Checking [Katoen, Thomas]
- Compiler Construction [Indermark, Noll]
- Semantics and Verification of Software [Noll]

Contact:

- Bastian Schlich (Inf. 11, `schlich@cs.rwth-aachen.de`)
- Thomas Noll (Inf. 2, `noll@cs.rwth-aachen.de`)

Outline