Formal Methods – An Appetizer

Chapter 4: Program Analysis

Flemming Nielson, Hanne Riis Nielson:

Formal Methods – An Appetizer.

ISBN 9783030051556, Springer 2019.

`FormalMethods.dk`

©Hanne Riis Nielson, Flemming Nielson, March 23, 2020.

Program Analysis

The Aim of Program Analysis

We want to provide guarantees about the behaviour of models and programs.

Fully Automatic

The technique is fully automatic so no user interventions needed.

Approximative

The technique works on approximate values, not the actual values.

Efficient

Highly efficient but sometimes at the price of lower precision.

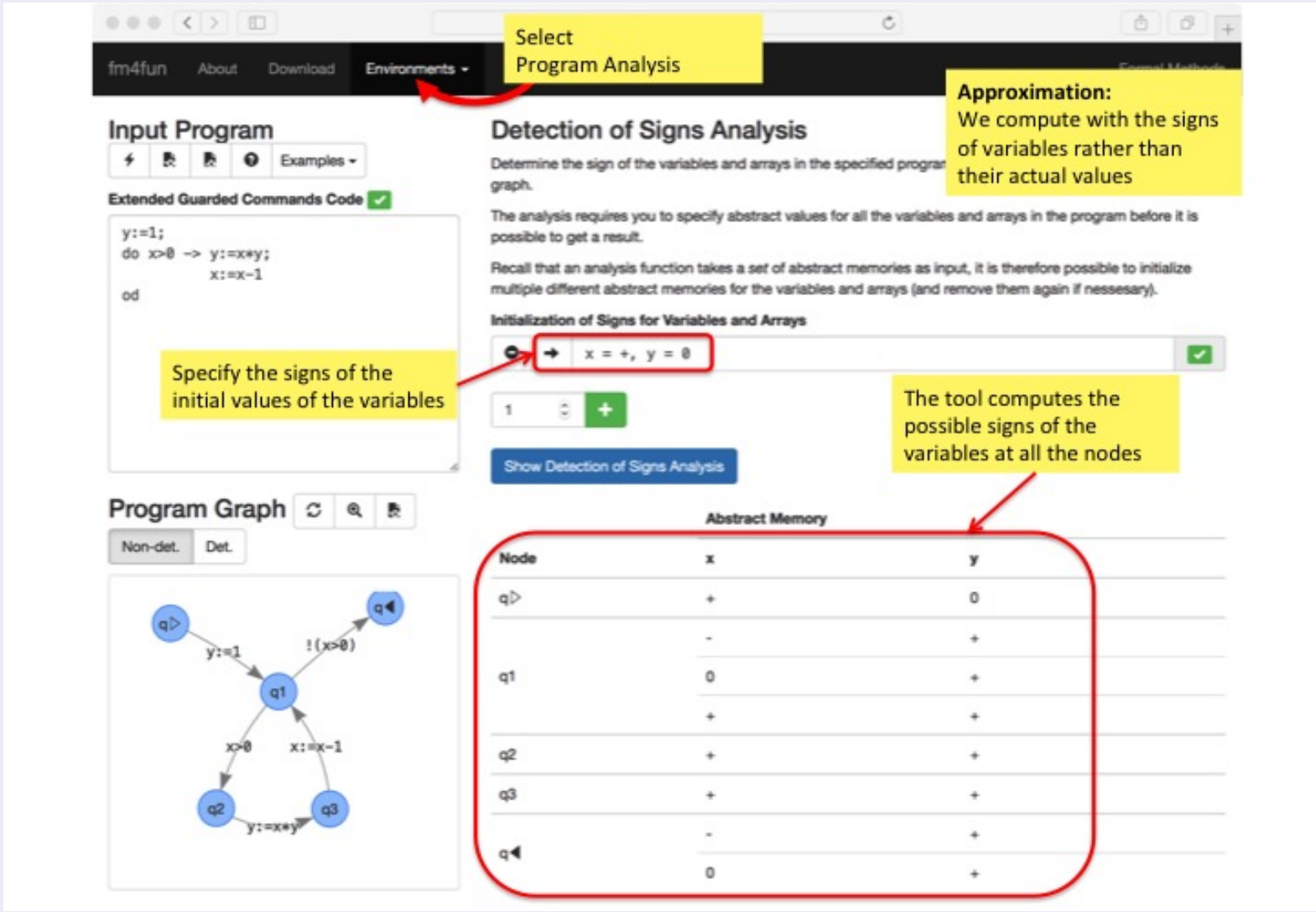
Application Area

Used in optimising compilers where they enable optimisations that for example reduce the run-time of programs.

Application Area

Used in software validation tools to pinpoint programming errors related to for example functional correctness and security.

FormalMethods.dk/fm4fun



Section 1

4.1 Abstract Properties

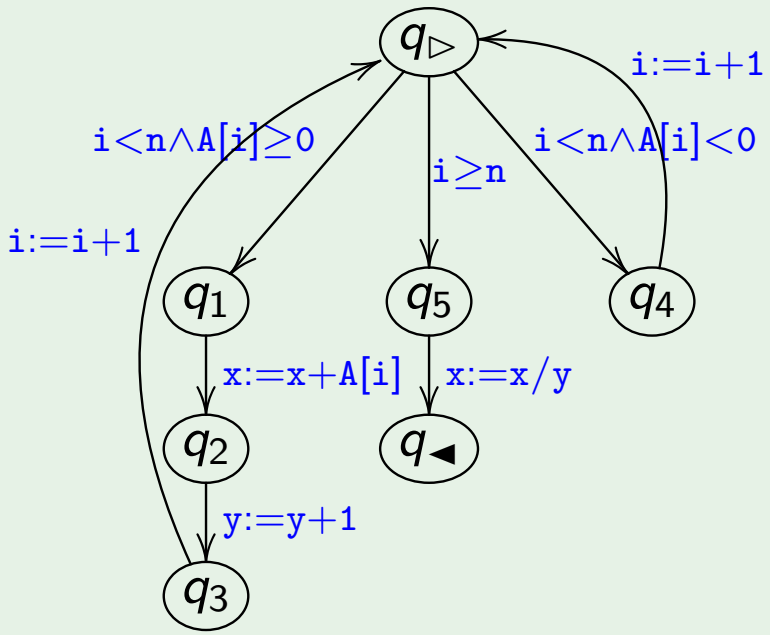
Flemming Nielson, Hanne Riis Nielson:
Formal Methods – An Appetizer.
ISBN 9783030051556, Springer 2019.

`FormalMethods.dk`

©Hanne Riis Nielson, Flemming Nielson, March 23, 2020.

Insights by Program Analysis

Program Graph (FM p 47)



Division by Zero? (FM p 47)

- if A only contains non-negative elements then we never divide by zero,
- if all the elements of A are negative then we will indeed divide by zero,
- in all other cases we may or may not divide by zero.

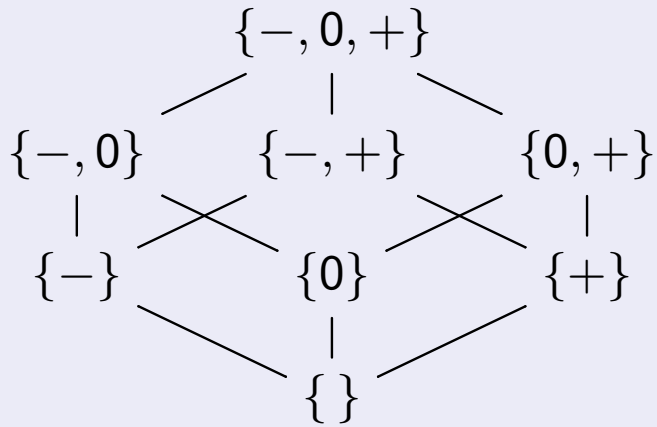
The aim is to develop a Detection of Signs analysis that will enable us to automatically compute this result.

Memories and Abstract Memories

The Abstract Properties: Signs

- for variables we record their *sign*, that is, an element of **Sign** $= \{-, 0, +\}$
- for arrays we record the *set of signs* of their entries, that is, an element of PowerSet(**Sign**)

PowerSet(**Sign**) (FM p 48)



Abstracting Memories (FM p 48)

x	0
y	1
i	1
n	3
A[0]	0
A[1]	4
A[2]	7

$\xrightarrow{\eta}$

x	0
y	+
i	+
n	+
A	$\{0, +\}$

Try It Out (FM p 48)

Give concrete memories corresponding to:

x	0
y	+
i	+
n	+
A	$\{+\}$

x	0
y	+
i	+
n	+
A	$\{-, 0, +\}$

The Abstraction of Memories

Memory

$$\sigma \in \mathbf{Mem} = \big(\ \mathbf{Var} \cup \{A[i] \mid A \in \mathbf{Arr}, 0 \leq i < \mathsf{size}(A)\} \ \big) \to \mathbf{Int}$$

Abstract Memory

$$(\hat{\sigma}_1, \hat{\sigma}_2) \in \widehat{\mathbf{Mem}} = (\mathbf{Var} \to \mathbf{Sign}) \times (\mathbf{Arr} \to \mathsf{PowerSet}(\mathbf{Sign}))$$

Abstraction: $\eta : \mathbf{Mem} \to \widehat{\mathbf{Mem}}$ (FM p 48)

Define $\eta(\sigma) = (\hat{\sigma}_1, \hat{\sigma}_2)$ where

for variables: $\hat{\sigma}_1(x) = \mathbf{sign}(v)$ where $v = \sigma(x)$

for arrays: $\hat{\sigma}_2(A) = \{\mathbf{sign}(v_1), \cdots, \mathbf{sign}(v_{k-1})\}$
where $v_0 = \sigma(A[0]), \cdots, v_{k-1} = \sigma(A[k-1])$,
and $k = \mathsf{length}(A)$

Section 2

4.2 Analysis Assignments

Flemming Nielson, Hanne Riis Nielson:
Formal Methods – An Appetizer.
ISBN 9783030051556, Springer 2019.

`FormalMethods.dk`

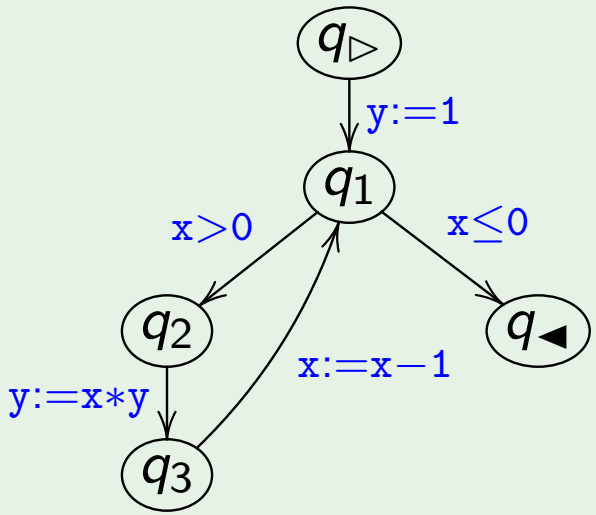
©Hanne Riis Nielson, Flemming Nielson, March 23, 2020.

Analysis Assignment

Analysis Assignment (FM p 49)

An *analysis assignment* of a program graph with nodes **Q** is a mapping

$$\mathbf{A} : \mathbf{Q} \to \text{PowerSet}(\widehat{\mathbf{Mem}})$$



Analysis Assignment for Factorial Program

$$\mathbf{A}(q_\triangleright) = \left\{ \begin{array}{|c|c|} \hline \mathtt{x} & + \\ \hline \mathtt{y} & 0 \\ \hline \end{array} \right\}$$

$$\mathbf{A}(q_1) = \left\{ \begin{array}{|c|c|} \hline \mathtt{x} & - \\ \hline \mathtt{y} & + \\ \hline \end{array}, \begin{array}{|c|c|} \hline \mathtt{x} & 0 \\ \hline \mathtt{y} & + \\ \hline \end{array}, \begin{array}{|c|c|} \hline \mathtt{x} & + \\ \hline \mathtt{y} & + \\ \hline \end{array} \right\}$$

$$\mathbf{A}(q_2) = \left\{ \begin{array}{|c|c|} \hline \mathtt{x} & + \\ \hline \mathtt{y} & + \\ \hline \end{array} \right\}$$

$$\mathbf{A}(q_3) = \left\{ \begin{array}{|c|c|} \hline \mathtt{x} & + \\ \hline \mathtt{y} & + \\ \hline \end{array} \right\}$$

$$\mathbf{A}(q_\blacktriangleleft) = \left\{ \begin{array}{|c|c|} \hline \mathtt{x} & - \\ \hline \mathtt{y} & + \\ \hline \end{array}, \begin{array}{|c|c|} \hline \mathtt{x} & 0 \\ \hline \mathtt{y} & + \\ \hline \end{array} \right\}$$

What Does Correctness Mean?

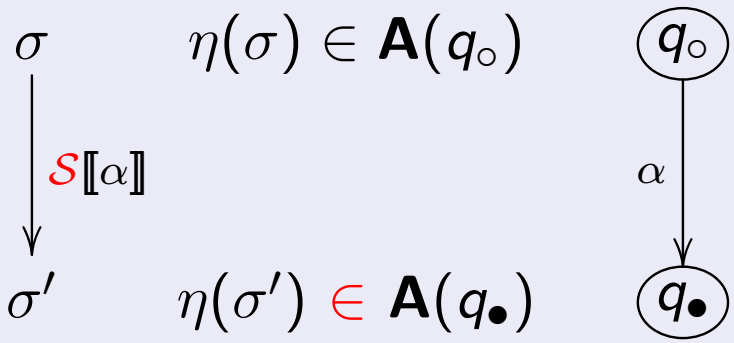
Definition of Correctness (FM p 50)

An analysis assignment $\mathbf{A} : \mathbf{Q} \to \text{PowerSet}(\widehat{\mathbf{Mem}})$ is *correct* if

- for all edges $(q_\circ, \alpha, q_\bullet)$ and for all memories σ:

$$\text{if} \quad \eta(\sigma) \in \mathbf{A}(q_\circ) \text{ and } \sigma' = \mathcal{S}[\![\alpha]\!]\sigma \quad \text{then} \quad \eta(\sigma') \in \mathbf{A}(q_\bullet)$$

- if $\sigma \in \mathbf{Mem}_\triangleright$ then $\eta(\sigma) \in \mathbf{A}(q_\triangleright)$; here $\mathbf{Mem}_\triangleright$ is the initial memories.



Proposition (FM p 50)

Let $\mathbf{A}$ be a correct analysis assignment and assume execution starts in $\langle q_\triangleright; \sigma \rangle$ where $\eta(\sigma) \in \mathbf{A}(q_\triangleright)$. If the execution reaches $\langle q; \sigma' \rangle$ then $\eta(\sigma') \in \mathbf{A}(q)$.

Idea: The analysis assignment abstracts the semantics of the program graph.

Hands On: Interpreting Analysis Assignments

FormalMethods.dk/fm4fun

Use the tool to analyse the program below with different sets of initial memories and use the correctness result to argue for the results listed.

Example Program

```
i:=0; x:=0; y:=0;
do (n>i) && (A[i]>=0) ->
        x:=x+A[i]; y:=y+1;
        i:=i+1
[] (n>i) && (0>A[i]) ->
        i:=i+1
od;
x:=x/y
```

Division by Zero? (FM p 47)

- if `A` only contains non-negative elements then we never divide by zero,
- if all the elements of `A` are negative then we will indeed divide by zero,
- in all other cases we may or may not divide by zero.

Section 3

4.3 Analysis Functions

Flemming Nielson, Hanne Riis Nielson:
Formal Methods – An Appetizer.
ISBN 9783030051556, Springer 2019.

`FormalMethods.dk`

©Hanne Riis Nielson, Flemming Nielson, March 23, 2020.

From Semantics to Computing with Signs

Semantics (Section 2.3)

Memory: $\mathbf{Mem} = \cdots$

$\mathcal{A}[\![a]\!] : \mathbf{Mem} \hookrightarrow \mathbf{Int}$

$\mathcal{B}[\![b]\!] : \mathbf{Mem} \hookrightarrow \mathbf{Bool}$

$\mathcal{S}[\![\alpha]\!] : \mathbf{Mem} \hookrightarrow \mathbf{Mem}$

Given a program graph, this gives rise to *execution sequences*.

Detection of Signs Analysis

Abstract Memory: $\widehat{\mathbf{Mem}} = \cdots$

$\widehat{\mathcal{A}}[\![a]\!] : \widehat{\mathbf{Mem}} \to \mathrm{PowerSet}(\mathbf{Sign})$

$\widehat{\mathcal{B}}[\![b]\!] : \widehat{\mathbf{Mem}} \to \mathrm{PowerSet}(\mathbf{Bool})$

$\widehat{\mathcal{S}}[\![\alpha]\!] : \mathrm{PowerSet}(\widehat{\mathbf{Mem}}) \to \mathrm{PowerSet}(\widehat{\mathbf{Mem}})$

Given a program graph, this gives rise to a *set of constraints*; the analysis assignment is a *solution* to these constraints.

Arithmetic Expressions: Computing with Signs

$\widehat{\mathcal{A}}[\![a]\!](\hat{\sigma}_1, \hat{\sigma}_2)$: Possible Signs of a in Abstract Memory $(\hat{\sigma}_1, \hat{\sigma}_2)$ (FM p 52)

$$\widehat{\mathcal{A}}[\![n]\!](\hat{\sigma}_1, \hat{\sigma}_2) = \{\mathbf{sign}(n)\}$$
$$\widehat{\mathcal{A}}[\![x]\!](\hat{\sigma}_1, \hat{\sigma}_2) = \{\hat{\sigma}_1(x)\}$$
$$\widehat{\mathcal{A}}[\![a_1 \ + \ a_2]\!](\hat{\sigma}_1, \hat{\sigma}_2) = \widehat{\mathcal{A}}[\![a_1]\!](\hat{\sigma}_1, \hat{\sigma}_2) \ \widehat{+} \ \widehat{\mathcal{A}}[\![a_2]\!](\hat{\sigma}_1, \hat{\sigma}_2)$$
$$\widehat{\mathcal{A}}[\![\cdots]\!](\hat{\sigma}_1, \hat{\sigma}_2) = \cdots$$

Adding Signs (FM p 52)

$\tilde{+}$	$-$	0	$+$
$-$	$\{-\}$	$\{-\}$	$\{-,0,+\}$
0	$\{-\}$	$\{0\}$	$\{+\}$
$+$	$\{-,0,+\}$	$\{+\}$	$\{+\}$

$$S_1 \ \widehat{op} \ S_2 = \bigcup_{s_1 \in S_1, s_2 \in S_2} s_1 \ \tilde{op} \ s_2$$

Try It Out (FM p 52)

Specify similar tables for $\tilde{-}$, $\tilde{*}$ and $\tilde{/}$.

Calculate $\{\}\widehat{+}\{0,+\}$, $\{0,+\}\widehat{-}\{0,+\}$, $\{0,+\}\widehat{*}\{0,+\}$ and $\{0,+\}\widehat{/}\{0,+\}$

Boolean Expressions: Computing with Signs

$\widehat{\mathcal{B}}[\![b]\!](\hat{\sigma}_1, \hat{\sigma}_2)$: Possible Truth Values of b in $(\hat{\sigma}_1, \hat{\sigma}_2)$ (FM p 53)

$$
\begin{aligned}
\widehat{\mathcal{B}}[\![\texttt{true}]\!](\hat{\sigma}_1, \hat{\sigma}_2) &= \{\mathrm{tt}\} \\
\widehat{\mathcal{B}}[\![a_1 \geq a_2]\!](\hat{\sigma}_1, \hat{\sigma}_2) &= \widehat{\mathcal{A}}[\![a_1]\!](\hat{\sigma}_1, \hat{\sigma}_2) \;\widehat{\geq}\; \widehat{\mathcal{A}}[\![a_2]\!](\hat{\sigma}_1, \hat{\sigma}_2) \\
\widehat{\mathcal{B}}[\![b_1 \wedge b_2]\!](\hat{\sigma}_1, \hat{\sigma}_2) &= \widehat{\mathcal{B}}[\![b_1]\!](\hat{\sigma}_1, \hat{\sigma}_2) \;\widehat{\wedge}\; \widehat{\mathcal{B}}[\![b_2]\!](\hat{\sigma}_1, \hat{\sigma}_2) \\
\widehat{\mathcal{B}}[\![\cdots]\!](\hat{\sigma}_1, \hat{\sigma}_2) &= \cdots
\end{aligned}
$$

Comparison (FM p 53)

$\tilde{\geq}$	$-$	0	$+$
$-$	$\{\mathrm{tt}, \mathrm{ff}\}$	$\{\mathrm{ff}\}$	$\{\mathrm{ff}\}$
0	$\{\mathrm{tt}\}$	$\{\mathrm{tt}\}$	$\{\mathrm{ff}\}$
$+$	$\{\mathrm{tt}\}$	$\{\mathrm{tt}\}$	$\{\mathrm{tt}, \mathrm{ff}\}$

Conjunction (FM p 53)

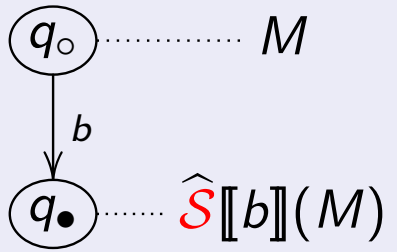
$\tilde{\wedge}$	tt	ff
tt	$\{\mathrm{tt}\}$	$\{\mathrm{ff}\}$
ff	$\{\mathrm{ff}\}$	$\{\mathrm{ff}\}$

Analysis of Tests and Assignments

Analysis of Test b (FM p 54)

Idea: keep all the abstract memories of M where b might hold:

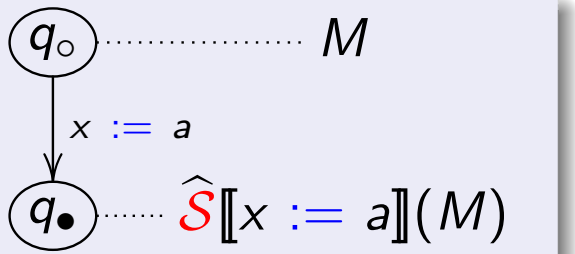
$$\widehat{\mathcal{S}}[\![b]\!](M) = \{(\hat{\sigma}_1, \hat{\sigma}_2) \mid (\hat{\sigma}_1, \hat{\sigma}_2) \in M \wedge \mathrm{tt} \in \widehat{\mathcal{B}}[\![b]\!](\hat{\sigma}_1, \hat{\sigma}_2)\}$$



Analysis of Assignment $x := a$ (FM p 54)

Idea: update all the abstract memories of M with the possible signs of a:

$$\widehat{\mathcal{S}}[\![x := a]\!](M) = \{(\hat{\sigma}_1[x \mapsto s], \hat{\sigma}_2) \mid (\hat{\sigma}_1, \hat{\sigma}_2) \in M \wedge s \in \widehat{\mathcal{A}}[\![a]\!](\hat{\sigma}_1, \hat{\sigma}_2)\}$$



Analysis of Array Lookup and Array Assignment

Possible Signs of The Arithmetic Expression $A[a]$ (FM p 52)

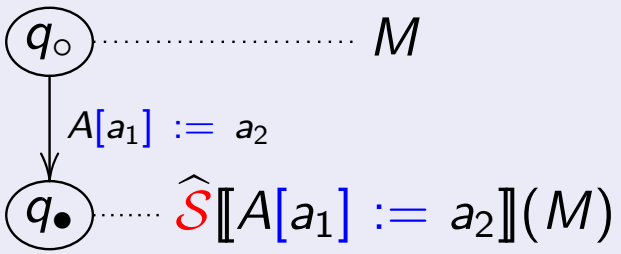
Idea: the lookup cannot be successful if a is negative:

$$\widehat{\mathcal{A}}[\![A[a]]\!](\hat{\sigma}_1, \hat{\sigma}_2) = \begin{cases} \hat{\sigma}_2(A) & \text{if } \widehat{\mathcal{A}}[\![a]]\!](\hat{\sigma}_1, \hat{\sigma}_2) \cap \{0, +\} \neq \{\} \\ \{\} & \text{otherwise} \end{cases}$$

Analysis of Assignment $A[a_1] := a_2$ (FM p 55)

Idea: update all the abstract memories of M with the possible sets of signs of the entries of A:

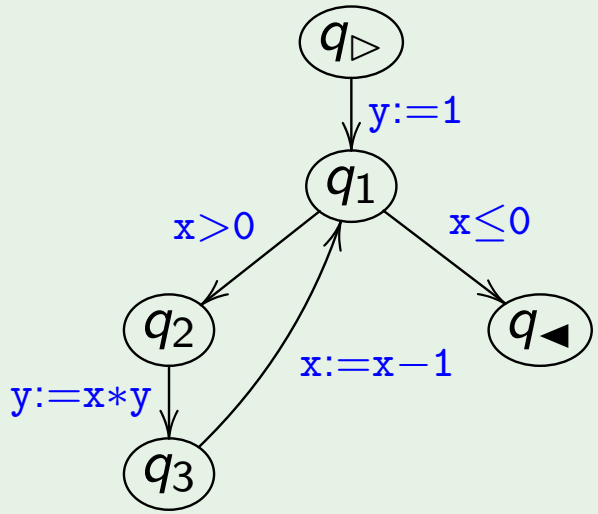
$$\begin{aligned} &\widehat{\mathcal{S}}[\![A[a_1] := a_2]\!](M) = \\ &\quad \{(\hat{\sigma}_1, \hat{\sigma}_2[A \mapsto S]) \mid (\hat{\sigma}_1, \hat{\sigma}_2) \in M \wedge \\ &\qquad \widehat{\mathcal{A}}[\![a_1]]\!](\hat{\sigma}_1, \hat{\sigma}_2) \cap \{0, +\} \neq \{\} \wedge \\ &\qquad \exists s' \in \hat{\sigma}_2(A) : \exists s'' \in \widehat{\mathcal{A}}[\![a_2]]\!](\hat{\sigma}_1, \hat{\sigma}_2) : \\ &\qquad\quad (\hat{\sigma}_2(A) \setminus \{s'\}) \cup \{s''\} \subseteq S \wedge \\ &\qquad\quad S \subseteq \hat{\sigma}_2(A) \cup \{s''\}\} \end{aligned}$$



Idea: s' might or might not be a posssible sign of A after the assignment; s'' is a possible sign of A after the assignment.

Try It Out: Checking Constraints

For each of the actions show that $\widehat{\mathcal{S}}[\![\alpha]\!](\mathbf{A}(q_\circ)) \sqsubseteq \mathbf{A}(q_\bullet)$



FormalMethods.dk/fm4fun

Perform similar checks on the analysis result for the program computing the average of some array entries.

Analysis Assignment for Factorial Program

$$\mathbf{A}(q_\triangleright) = \left\{ \begin{array}{|c|c|}\hline \mathtt{x} & + \\ \hline \mathtt{y} & 0 \\ \hline \end{array} \right\}$$

$$\mathbf{A}(q_1) = \left\{ \begin{array}{|c|c|}\hline \mathtt{x} & - \\ \hline \mathtt{y} & + \\ \hline \end{array}, \begin{array}{|c|c|}\hline \mathtt{x} & 0 \\ \hline \mathtt{y} & + \\ \hline \end{array}, \begin{array}{|c|c|}\hline \mathtt{x} & + \\ \hline \mathtt{y} & + \\ \hline \end{array} \right\}$$

$$\mathbf{A}(q_2) = \left\{ \begin{array}{|c|c|}\hline \mathtt{x} & + \\ \hline \mathtt{y} & + \\ \hline \end{array} \right\}$$

$$\mathbf{A}(q_3) = \left\{ \begin{array}{|c|c|}\hline \mathtt{x} & + \\ \hline \mathtt{y} & + \\ \hline \end{array} \right\}$$

$$\mathbf{A}(q_\blacktriangleleft) = \left\{ \begin{array}{|c|c|}\hline \mathtt{x} & - \\ \hline \mathtt{y} & + \\ \hline \end{array}, \begin{array}{|c|c|}\hline \mathtt{x} & 0 \\ \hline \mathtt{y} & + \\ \hline \end{array} \right\}$$

Section 4

4.4 Analysis Specification

Flemming Nielson, Hanne Riis Nielson:
Formal Methods – An Appetizer.
ISBN 9783030051556, Springer 2019.

`FormalMethods.dk`

©Hanne Riis Nielson, Flemming Nielson, March 23, 2020.

Can We Trust the Analysis Results?

Key Idea

We want to ensure *once for all* that the analysis assignment **A** computed by the analysis for a program graph **PG** is *correct*.

If so, then any execution starting in $\langle q_\triangleright; \sigma \rangle$ where $\eta(\sigma) \in \mathbf{A}(q_\triangleright)$ will satisfy that if it reaches $\langle q; \sigma' \rangle$ then $\eta(\sigma') \in \mathbf{A}(q)$.

Definition of Correctness (FM p 50)

Recall that an analysis assignment $\mathbf{A} : \mathbf{Q} \to \text{PowerSet}(\widehat{\mathbf{Mem}})$ is *correct* if

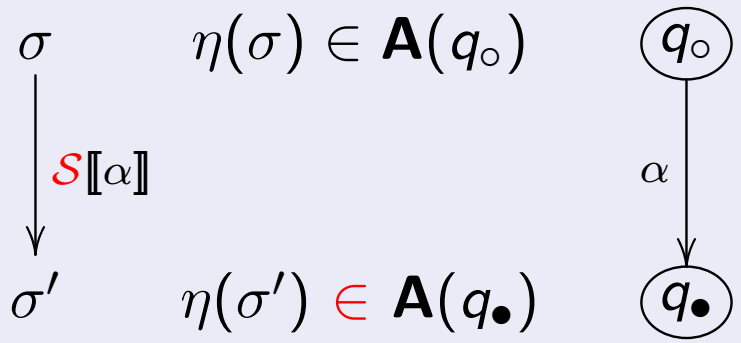
- for all edges $(q_\circ, \alpha, q_\bullet)$ and for all memories σ:

  if $\eta(\sigma) \in \mathbf{A}(q_\circ)$ and $\sigma' = \mathcal{S}[\![\alpha]\!]\sigma$ then $\eta(\sigma') \in \mathbf{A}(q_\bullet)$

- if $\sigma \in \mathbf{Mem}_\triangleright$ then $\eta(\sigma) \in \mathbf{A}(q_\triangleright)$; here $\mathbf{Mem}_\triangleright$ is the initial memories.

How to Ensure This?

- The program analysis must be *sound* with respect to the semantics
- The analysis assignment must be a *solution* for the program graph



Specification of Program Analysis

A Program Analysis is Specified By (FM p 55)

- a finite analysis domain $(\text{PowerSet}(\widehat{\textbf{Mem}}), \subseteq)$
- a *monotonic* analysis function for each action α: $\widehat{\mathcal{S}}[\![\alpha]\!] : \text{PowerSet}(\widehat{\textbf{Mem}}) \to \text{PowerSet}(\widehat{\textbf{Mem}})$
- a set $\widehat{\textbf{Mem}}_{\triangleright}$ of initial abstract memories

The function f is *monotonic* if

$$M_1 \subseteq M_2$$

implies

$$f(M_1) \subseteq f(M_2)$$

for all M_1, M_2

Try It Out (FM p 55)

Show that the analysis functions for the detection of signs analysis are monotonic.

$$\widehat{\mathcal{S}}[\![A[a_1] := a_2]\!](M) = \cdots$$

$$\widehat{\mathcal{S}}[\![b]\!](M) = \{(\hat{\sigma}_1, \hat{\sigma}_2) \mid (\hat{\sigma}_1, \hat{\sigma}_2) \in M \wedge \text{tt} \in \widehat{\mathcal{B}}[\![b]\!](\hat{\sigma}_1, \hat{\sigma}_2)\}$$

$$\widehat{\mathcal{S}}[\![x := a]\!](M) = \{(\hat{\sigma}_1[x \mapsto s], \hat{\sigma}_2) \mid (\hat{\sigma}_1, \hat{\sigma}_2) \in M \wedge s \in \widehat{\mathcal{A}}[\![a]\!](\hat{\sigma}_1, \hat{\sigma}_2)\}$$

Soundness of a Program Analysis

Soundness (FM p 56)

A program analysis is sound with respect to the semantics if

- if $\sigma' = \mathcal{S}[\![\alpha]\!](\sigma)$ and $\eta(\sigma) \in M$ for some set M then $\eta(\sigma') \in \widehat{\mathcal{S}}[\![\alpha]\!](M)$, and
- if $\sigma \in \mathbf{Mem}_\triangleright$ then $\eta(\sigma) \in \widehat{\mathbf{Mem}}_\triangleright$.

$$\begin{array}{ccc} \sigma & \eta(\sigma) & \in & M \\ \downarrow \mathcal{S}[\![\alpha]\!] & & & \downarrow \widehat{\mathcal{S}}[\![\alpha]\!] \\ \sigma' & \eta(\sigma') & \in & M' \end{array}$$

Soundness Result (FM p 56)

The *detection of signs* analysis is sound provided that:

$$\sigma \in \mathbf{Mem}_\triangleright \;\Rightarrow\; \eta(\sigma) \in \widehat{\mathbf{Mem}}_\triangleright$$

For Arithmetic Expressions (FM p 52)

If $v = \mathcal{A}[\![a]\!]\sigma$ is defined then $\mathbf{sign}(v) \in \widehat{\mathcal{A}}[\![a]\!](\eta(\sigma))$

For Boolean Expressions (FM p 53)

If $\mathcal{B}[\![b]\!]\sigma$ is defined then $\mathcal{B}[\![b]\!]\sigma \in \widehat{\mathcal{B}}[\![b]\!](\eta(\sigma))$

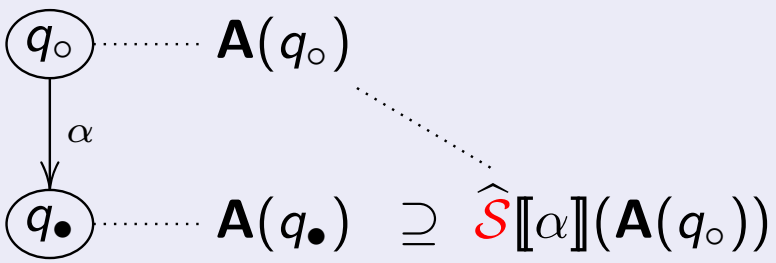
A soundness result is proved once for all when a program analysis is specified.

An Analysis Assignment is a Solution

Solutions (FM p 57)

An analysis assignment $\mathbf{A} : \mathbf{Q} \to \text{PowerSet}(\widehat{\mathbf{Mem}})$ is a solution for a program graph **PG** if

- for all edges $(q_\circ, \alpha, q_\bullet)$ we have $\widehat{\mathcal{S}}[\![\alpha]\!](\mathbf{A}(q_\circ)) \subseteq \mathbf{A}(q_\bullet)$, and
- $\widehat{\mathbf{Mem}}_\triangleright \subseteq \mathbf{A}(q_\triangleright)$



Idea: An analysis assignment is a solution if it solves a set of constraints constructed from the program graph

Constraints for Factorial Program

$$\left\{\begin{array}{|c|c|}\hline \mathtt{x} & + \\ \hline \mathtt{y} & 0 \\ \hline \end{array}\right\} \subseteq \mathbf{A}(q_\triangleright)$$

$$\widehat{\mathcal{S}}[\![\mathtt{y} := 1]\!](\mathbf{A}(q_\triangleright)) \subseteq \mathbf{A}(q_1)$$

$$\widehat{\mathcal{S}}[\![\mathtt{x} > 0]\!](\mathbf{A}(q_1)) \subseteq \mathbf{A}(q_2)$$

$$\widehat{\mathcal{S}}[\![\mathtt{y} := \mathtt{x} * \mathtt{y}]\!](\mathbf{A}(q_2)) \subseteq \mathbf{A}(q_3)$$

$$\widehat{\mathcal{S}}[\![\mathtt{x} := \mathtt{x} - 1]\!](\mathbf{A}(q_3)) \subseteq \mathbf{A}(q_1)$$

$$\widehat{\mathcal{S}}[\![\mathtt{x} \leq 0]\!](\mathbf{A}(q_1)) \subseteq \mathbf{A}(q_\blacktriangleleft)$$

Summary of Results

Sound Analysis Specification (FM p 56)

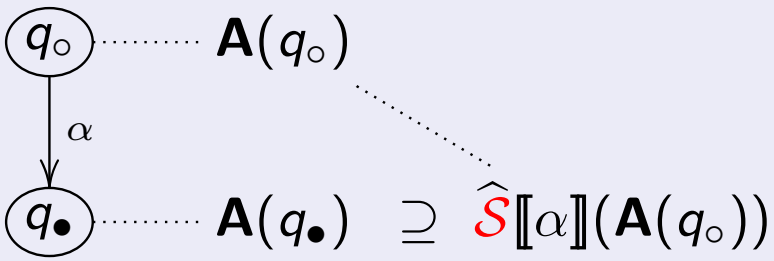
$$\begin{array}{ccc} \sigma & \eta(\sigma) \in & M \\ \downarrow \mathcal{S}[\![\alpha]\!] & & \downarrow \widehat{\mathcal{S}}[\![\alpha]\!] \\ \sigma' & \eta(\sigma') \in & M' \end{array}$$

$\sigma \in \mathbf{Mem}_\triangleright$ implies $\eta(\sigma) \in \widehat{\mathbf{Mem}}_\triangleright$

Proposition (FM p 58)

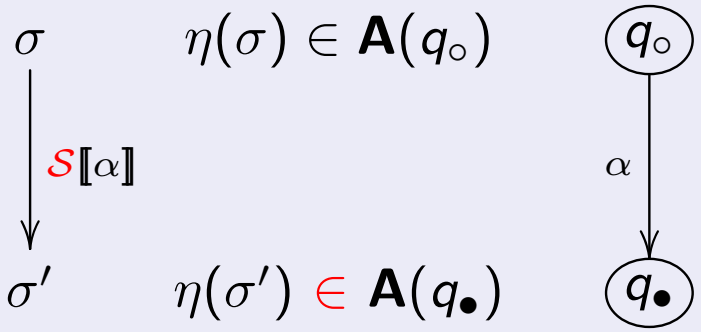
For a sound analysis specification all analysis assignments that are solutions are also correct.

Solution (FM p 57)



$\widehat{\mathbf{Mem}}_\triangleright \subseteq \mathbf{A}(q_\triangleright)$

Correct Analysis Assignment (FM p 50)



$\sigma \in \mathbf{Mem}_\triangleright$ implies $\eta(\sigma) \in \mathbf{A}(q_\triangleright)$

Section 5

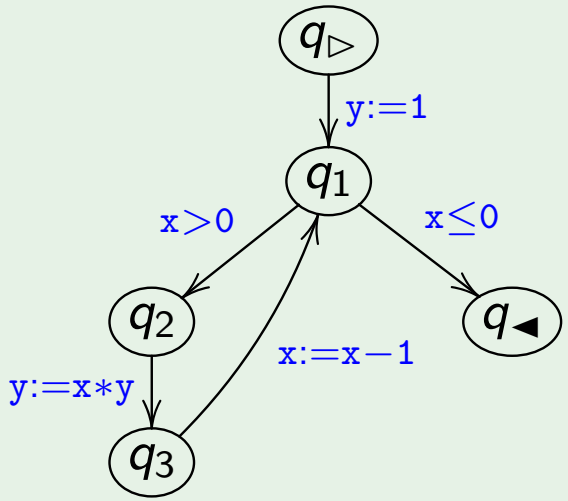
4.5 Computing Solutions

Flemming Nielson, Hanne Riis Nielson:
Formal Methods – An Appetizer.
ISBN 9783030051556, Springer 2019.

`FormalMethods.dk`

©Hanne Riis Nielson, Flemming Nielson, March 23, 2020.

Example: Factorial Program



Idea: Compute an analysis assignment that solves the constraints obtained from the program graph

$$\left\{\begin{array}{|c|c|}\hline \mathtt{x} & + \\ \hline \mathtt{y} & 0 \\ \hline\end{array}\right\} \subseteq \mathbf{A}(q_\triangleright)$$

$$\widehat{\mathcal{S}}[\![\mathtt{y} := 1]\!](\mathbf{A}(q_\triangleright)) \subseteq \mathbf{A}(q_1)$$

$$\widehat{\mathcal{S}}[\![\mathtt{x} > 0]\!](\mathbf{A}(q_1)) \subseteq \mathbf{A}(q_2)$$

$$\widehat{\mathcal{S}}[\![\mathtt{y} := \mathtt{x} * \mathtt{y}]\!](\mathbf{A}(q_2)) \subseteq \mathbf{A}(q_3)$$

$$\widehat{\mathcal{S}}[\![\mathtt{x} := \mathtt{x} - 1]\!](\mathbf{A}(q_3)) \subseteq \mathbf{A}(q_1)$$

$$\widehat{\mathcal{S}}[\![\mathtt{x} \leq 0]\!](\mathbf{A}(q_1)) \subseteq \mathbf{A}(q_\blacktriangleleft)$$

$$\mathbf{A}(q_\triangleright) = \left\{\begin{array}{|c|c|}\hline \mathtt{x} & + \\ \hline \mathtt{y} & 0 \\ \hline\end{array}\right\}$$

$$\mathbf{A}(q_1) = \left\{\begin{array}{|c|c|}\hline \mathtt{x} & - \\ \hline \mathtt{y} & + \\ \hline\end{array}, \begin{array}{|c|c|}\hline \mathtt{x} & 0 \\ \hline \mathtt{y} & + \\ \hline\end{array}, \begin{array}{|c|c|}\hline \mathtt{x} & + \\ \hline \mathtt{y} & + \\ \hline\end{array}\right\}$$

$$\mathbf{A}(q_2) = \left\{\begin{array}{|c|c|}\hline \mathtt{x} & + \\ \hline \mathtt{y} & + \\ \hline\end{array}\right\}$$

$$\mathbf{A}(q_3) = \left\{\begin{array}{|c|c|}\hline \mathtt{x} & + \\ \hline \mathtt{y} & + \\ \hline\end{array}\right\}$$

$$\mathbf{A}(q_\blacktriangleleft) = \left\{\begin{array}{|c|c|}\hline \mathtt{x} & - \\ \hline \mathtt{y} & + \\ \hline\end{array}, \begin{array}{|c|c|}\hline \mathtt{x} & 0 \\ \hline \mathtt{y} & + \\ \hline\end{array}\right\}$$

Algorithm for Computing Solutions

Chaotic Iteration (FM p 59)

forall $q \in \mathbf{Q} \setminus \{q_\triangleright\}$ do $\mathbf{A}(q) := \{\}$;

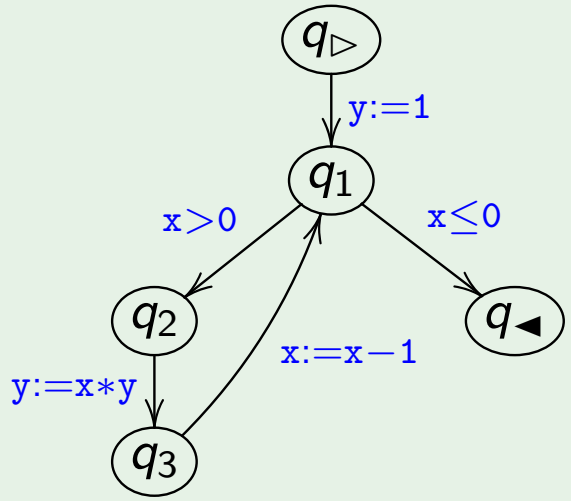
$\mathbf{A}(q_\triangleright) := \widehat{\mathbf{Mem}}_\triangleright$;

while there exists an edge $(q_\circ, \alpha, q_\bullet) \in \mathbf{E}$
such that $\widehat{\mathcal{S}}[\![\alpha]\!](\mathbf{A}(q_\circ)) \not\subseteq \mathbf{A}(q_\bullet)$

do $\mathbf{A}(q_\bullet) := \mathbf{A}(q_\bullet) \cup \widehat{\mathcal{S}}[\![\alpha]\!](\mathbf{A}(q_\circ))$

Proposition (FM p 59)

The algorithm always terminates and it returns an analysis assignment that is a solution to the analysis problem.



Try It Out: Factorial Program

Use the algorithm to compute an analysis assignment for the factorial program in the case where

$$\widehat{\mathbf{Mem}}_\triangleright = \left\{ \begin{array}{|c|c|} \hline \mathtt{x} & + \\ \hline \mathtt{y} & 0 \\ \hline \end{array} \right\}$$

Refinements of The Algorithm

FormalMethods.dk/fm4fun

The tool implements a deterministic version of the chaotic iteration algorithm using a worklist to keep track of the nodes to consider.

Worklist Algorithm (FM p 60)

forall $q \in \mathbf{Q} \setminus \{q_\triangleright\}$ do $\mathbf{A}(q) := \{\}$;
$\mathbf{A}(q_\triangleright) := \widehat{\mathbf{Mem}_\triangleright}$;
$\mathbf{W} := \{q_\triangleright\}$;

while $\mathbf{W} \neq \{\}$ do
  choose $q_\circ \in \mathbf{W}$; $\mathbf{W} := \mathbf{W} \setminus \{q_\circ\}$;
  for all edges $(q_\circ, \alpha, q_\bullet) \in \mathbf{E}$ do
    if $\widehat{\mathcal{S}}[\![\alpha]\!](\mathbf{A}(q_\circ)) \not\subseteq \mathbf{A}(q_\bullet)$
    then $\mathbf{A}(q_\bullet) := \mathbf{A}(q_\bullet) \cup \widehat{\mathcal{S}}[\![\alpha]\!](\mathbf{A}(q_\circ))$;
      $\mathbf{W} := \mathbf{W} \cup \{q_\bullet\}$;

Organising the Worklist

The worklist $\mathbf{W}$ can be implemented as a queue:

- $q_\circ$ is taken from the front of the queue
- $q_\bullet$ is inserted in the rear of the queue

The efficiency of the algorithm depends on how the worklist is organised