ISR 2026 — Basic Track

Jörg Endrullis
Cynthia Kop
Femke van Raamsdonk

partially based on slides by: Aart Middeldorp

hosting institution:
Radboud Universiteit Nijmegen
The Netherlands

Recall: ARSs

Recall: an ARS is a pair $(A, \to)$ of a set and a reduction relation.

Recall: ARSs

Recall: an ARS is a pair $(A, \to)$ of a set and a reduction relation.

Term rewriting: $(\mathcal{T}(\Sigma, \mathcal{X}), \to_{\mathcal{R}})$

Varying the set of terms

Example:

$$\begin{aligned} \mathsf{add}(0, y) &\to y \\ \mathsf{add}(\mathsf{s}(x), y) &\to \mathsf{add}(x, \mathsf{s}(y)) \\ \mathsf{cost}(\mathsf{apple}) &\to \mathsf{s}(0) \\ \mathsf{cost}(\mathsf{pear}) &\to \mathsf{s}(\mathsf{s}(0)) \end{aligned}$$

Varying the set of terms

Example:

$$\begin{aligned} \mathsf{add}(0, y) &\to y \\ \mathsf{add}(\mathsf{s}(x), y) &\to \mathsf{add}(x, \mathsf{s}(y)) \\ \mathsf{cost}(\mathsf{apple}) &\to \mathsf{s}(0) \\ \mathsf{cost}(\mathsf{pear}) &\to \mathsf{s}(\mathsf{s}(0)) \end{aligned}$$

We can create:

$$\mathsf{add}(0, \mathsf{apple})$$

Varying the set of terms

Example:

$$\begin{aligned} \mathsf{add}(0, y) &\to y \\ \mathsf{add}(\mathsf{s}(x), y) &\to \mathsf{add}(x, \mathsf{s}(y)) \\ \mathsf{cost}(\mathsf{apple}) &\to \mathsf{s}(0) \\ \mathsf{cost}(\mathsf{pear}) &\to \mathsf{s}(\mathsf{s}(0)) \end{aligned}$$

We can create:

$$\mathsf{add}(0, \mathsf{apple})$$

Another example:

$$\begin{aligned} \mathsf{length}(\mathsf{nil}) &\to 0 \\ \mathsf{length}(\mathsf{cons}(x, y)) &\to \mathsf{s}(\mathsf{length}(y)) \end{aligned}$$

Varying the set of terms

Example:

$$\begin{aligned} \mathsf{add}(0, y) &\to y \\ \mathsf{add}(\mathsf{s}(x), y) &\to \mathsf{add}(x, \mathsf{s}(y)) \\ \mathsf{cost}(\mathsf{apple}) &\to \mathsf{s}(0) \\ \mathsf{cost}(\mathsf{pear}) &\to \mathsf{s}(\mathsf{s}(0)) \end{aligned}$$

We can create:

$$\mathsf{add}(0, \mathsf{apple})$$

Another example:

$$\begin{aligned} \mathsf{length}(\mathsf{nil}) &\to 0 \\ \mathsf{length}(\mathsf{cons}(x, y)) &\to \mathsf{s}(\mathsf{length}(y)) \end{aligned}$$

What is the meaning of: $\mathsf{length}(\mathsf{s}(\mathsf{cons}(\mathsf{nil}, 0)))$?

Problems of type-insensitive analysis

$$
\begin{aligned}
\mathsf{f}(0) &\to 0 \\
\mathsf{f}(\mathsf{s}(x)) &\to 0 \\
\mathsf{cost}(\mathsf{apple}) &\to \mathsf{s}(0) \\
\mathsf{cost}(\mathsf{pear}) &\to \mathsf{s}(\mathsf{s}(0))
\end{aligned}
$$

Problems of type-insensitive analysis

$$
\begin{aligned}
\mathsf{f}(0) &\to 0 \\
\mathsf{f}(\mathsf{s}(x)) &\to 0 \\
\mathsf{cost}(\mathsf{apple}) &\to \mathsf{s}(0) \\
\mathsf{cost}(\mathsf{pear}) &\to \mathsf{s}(\mathsf{s}(0))
\end{aligned}
$$

Question: are the following properties satisfied?

Problems of type-insensitive analysis

$$\begin{aligned} \mathsf{f}(0) &\to 0 \\ \mathsf{f}(\mathsf{s}(x)) &\to 0 \\ \mathsf{cost}(\mathsf{apple}) &\to \mathsf{s}(0) \\ \mathsf{cost}(\mathsf{pear}) &\to \mathsf{s}(\mathsf{s}(0)) \end{aligned}$$

Question: are the following properties satisfied?

- $\mathsf{f}(t)$ reduces to 0 for all t

Problems of type-insensitive analysis

$$\begin{aligned} \mathsf{f}(0) &\to 0 \\ \mathsf{f}(\mathsf{s}(x)) &\to 0 \\ \mathsf{cost}(\mathsf{apple}) &\to \mathsf{s}(0) \\ \mathsf{cost}(\mathsf{pear}) &\to \mathsf{s}(\mathsf{s}(0)) \end{aligned}$$

Question: are the following properties satisfied?

- $\mathsf{f}(t)$ reduces to 0 for all t
- every ground term reduces to a "constructor normal form"

Problems of type-insensitive analysis

$$\begin{aligned} \mathsf{f}(0) &\to 0 \\ \mathsf{f}(\mathsf{s}(x)) &\to 0 \\ \mathsf{cost}(\mathsf{apple}) &\to \mathsf{s}(0) \\ \mathsf{cost}(\mathsf{pear}) &\to \mathsf{s}(\mathsf{s}(0)) \end{aligned}$$

Question: are the following properties satisfied?

- $\mathsf{f}(t)$ reduces to 0 for all t
- every ground term reduces to a "constructor normal form"

Termination: Toyama's example:

$$\begin{aligned} \mathsf{f}(x,1,2) &\to \mathsf{f}(x,x,x) \\ \mathsf{chooselist}(x,y) &\to x \\ \mathsf{chooselist}(x,y) &\to y \end{aligned}$$

Typing a term rewriting system (intuition)

Idea: add types to function symbols

Typing a term rewriting system (intuition)

Idea: add types to function symbols

Example: $0 :: \mathsf{nat}$, $\mathsf{apple} :: \mathsf{fruit}$, $\mathsf{banana} :: \mathsf{fruit}$, $\mathsf{s} :: \mathsf{nat} \Rightarrow \mathsf{nat}$, $\mathsf{cost} :: \mathsf{fruit} \Rightarrow \mathsf{nat}$, $\mathsf{add} :: \mathsf{nat} \Rightarrow \mathsf{nat} \Rightarrow \mathsf{nat}$, $\mathsf{nil} :: \mathsf{list}$, $\mathsf{cons} :: \mathsf{nat} \Rightarrow \mathsf{list} \Rightarrow \mathsf{list}$

Typing a term rewriting system (intuition)

Idea: add types to function symbols

Example: $0 :: \mathsf{nat}$, $\mathsf{apple} :: \mathsf{fruit}$, $\mathsf{banana} :: \mathsf{fruit}$, $\mathsf{s} :: \mathsf{nat} \Rightarrow \mathsf{nat}$, $\mathsf{cost} :: \mathsf{fruit} \Rightarrow \mathsf{nat}$, $\mathsf{add} :: \mathsf{nat} \Rightarrow \mathsf{nat} \Rightarrow \mathsf{nat}$, $\mathsf{nil} :: \mathsf{list}$, $\mathsf{cons} :: \mathsf{nat} \Rightarrow \mathsf{list} \Rightarrow \mathsf{list}$

Requirement: terms must be well-typed!

Typing a term rewriting system (intuition)

Idea: add types to function symbols

Example: $0 :: \mathsf{nat}$, $\mathsf{apple} :: \mathsf{fruit}$, $\mathsf{banana} :: \mathsf{fruit}$, $\mathsf{s} :: \mathsf{nat} \Rightarrow \mathsf{nat}$, $\mathsf{cost} :: \mathsf{fruit} \Rightarrow \mathsf{nat}$, $\mathsf{add} :: \mathsf{nat} \Rightarrow \mathsf{nat} \Rightarrow \mathsf{nat}$, $\mathsf{nil} :: \mathsf{list}$, $\mathsf{cons} :: \mathsf{nat} \Rightarrow \mathsf{list} \Rightarrow \mathsf{list}$

Requirement: terms must be well-typed!

- Terms: $\mathsf{s}(\mathsf{add}(0, 0))$ and $\mathsf{cons}(0, \mathsf{cons}(\mathsf{add}(0, \mathsf{cost}(\mathsf{apple})), \mathsf{nil}))$
- Not terms: $\mathsf{cons}(0, \mathsf{banana})$ and $\mathsf{s}(\mathsf{nil})$

Typing a term rewriting system (intuition)

Idea: add types to function symbols

Example: $0 :: \mathsf{nat}$, $\mathsf{apple} :: \mathsf{fruit}$, $\mathsf{banana} :: \mathsf{fruit}$, $\mathsf{s} :: \mathsf{nat} \Rightarrow \mathsf{nat}$, $\mathsf{cost} :: \mathsf{fruit} \Rightarrow \mathsf{nat}$, $\mathsf{add} :: \mathsf{nat} \Rightarrow \mathsf{nat} \Rightarrow \mathsf{nat}$, $\mathsf{nil} :: \mathsf{list}$, $\mathsf{cons} :: \mathsf{nat} \Rightarrow \mathsf{list} \Rightarrow \mathsf{list}$

Requirement: terms must be well-typed!

- Terms: $\mathsf{s}(\mathsf{add}(0, 0))$ and $\mathsf{cons}(0, \mathsf{cons}(\mathsf{add}(0, \mathsf{cost}(\mathsf{apple})), \mathsf{nil}))$
- Not terms: $\mathsf{cons}(0, \mathsf{banana})$ and $\mathsf{s}(\mathsf{nil})$

Variables: implicitly carry a type as well (but usually not mentioned)

- Allowed: $\mathsf{add}(x, x)$
- Not allowed: $\mathsf{cons}(x, x)$

Typing a term rewriting system (intuition)

Idea: add types to function symbols

Example: $0 :: \mathsf{nat}$, $\mathsf{apple} :: \mathsf{fruit}$, $\mathsf{banana} :: \mathsf{fruit}$, $\mathsf{s} :: \mathsf{nat} \Rightarrow \mathsf{nat}$, $\mathsf{cost} :: \mathsf{fruit} \Rightarrow \mathsf{nat}$, $\mathsf{add} :: \mathsf{nat} \Rightarrow \mathsf{nat} \Rightarrow \mathsf{nat}$, $\mathsf{nil} :: \mathsf{list}$, $\mathsf{cons} :: \mathsf{nat} \Rightarrow \mathsf{list} \Rightarrow \mathsf{list}$

Requirement: terms must be well-typed!

- Terms: $\mathsf{s}(\mathsf{add}(0, 0))$ and $\mathsf{cons}(0, \mathsf{cons}(\mathsf{add}(0, \mathsf{cost}(\mathsf{apple})), \mathsf{nil}))$
- Not terms: $\mathsf{cons}(0, \mathsf{banana})$ and $\mathsf{s}(\mathsf{nil})$

Variables: implicitly carry a type as well (but usually not mentioned)

- Allowed: $\mathsf{add}(x, x)$
- Not allowed: $\mathsf{cons}(x, x)$

Reduction: unchanged!

Definition

- signature Σ function symbols $\mathsf{f} \in \Sigma$ with arities $\#(f)$
- variables $\mathcal{X}$ $\Sigma \cap \mathcal{X} = \varnothing$ infinitely many

- terms $\mathcal{T}(\Sigma, \mathcal{X})$ smallest set such that
  - $\mathcal{X} \subseteq \mathcal{T}(\Sigma, \mathcal{X})$
  - if $\mathsf{f} \in \Sigma$ has arity $n \geqslant 0$ and $t_1, \ldots, t_n \in \mathcal{T}(\Sigma, \mathcal{X})$ then $\mathsf{f}(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma, \mathcal{X})$

Definition

- signature Σ function symbols $\mathsf{f} \in \Sigma$ ~~with arities $\#(f)$~~
- variables $\mathcal{X}$ $\Sigma \cap \mathcal{X} = \varnothing$ infinitely many
- type function typeof maps each f to a type $\sigma_1 \Rightarrow \cdots \Rightarrow \sigma_n \Rightarrow \tau$ and each x to a sort
- terms $\mathcal{T}(\Sigma, \mathcal{X})$ smallest set such that
  - $\mathcal{X} \subseteq \mathcal{T}(\Sigma, \mathcal{X})$
  - if $\mathsf{f} \in \Sigma$ has arity $n \geqslant 0$ and $t_1, \ldots, t_n \in \mathcal{T}(\Sigma, \mathcal{X})$ then $\mathsf{f}(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma, \mathcal{X})$

Definition

- signature Σ function symbols $\mathsf{f} \in \Sigma$ ~~with arities $\#(f)$~~
- variables $\mathcal{X}$ $\Sigma \cap \mathcal{X} = \varnothing$ infinitely many
- type function typeof maps each f to a type $\sigma_1 \Rightarrow \cdots \Rightarrow \sigma_n \Rightarrow \tau$ and each x to a sort
- terms $\mathcal{T}(\Sigma, \mathcal{X})$ smallest set ~~such that~~ typable under:
  - $\mathcal{X} \subseteq \mathcal{T}(\Sigma, \mathcal{X})$
  - if $\mathsf{f} \in \Sigma$ has arity $n \geqslant 0$ and $t_1, \ldots, t_n \in \mathcal{T}(\Sigma, \mathcal{X})$ then $\mathsf{f}(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma, \mathcal{X})$

Definition

- signature Σ function symbols $\mathsf{f} \in \Sigma$ ~~with arities $\#(f)$~~
- variables $\mathcal{X}$ $\Sigma \cap \mathcal{X} = \varnothing$ infinitely many
- type function typeof maps each f to a type $\sigma_1 \Rightarrow \cdots \Rightarrow \sigma_n \Rightarrow \tau$ and each x to a sort
- terms $\mathcal{T}(\Sigma, \mathcal{X})$ smallest set ~~such that~~ typable under:
  - ~~$\mathcal{X} \subseteq \mathcal{T}(\Sigma, \mathcal{X})$~~
    $x :: \sigma$ if $x \in \mathcal{X}$ and $\mathrm{typeof}(x) = \sigma$
  - ~~if $\mathsf{f} \in \Sigma$ has arity $n \geqslant 0$ and $t_1, \ldots, t_n \in \mathcal{T}(\Sigma, \mathcal{X})$ then $\mathsf{f}(t_1, \ldots, t_n) \in \mathcal{T}(\Sigma, \mathcal{X})$~~
    if $\mathrm{typeof}(\mathsf{f}) = \sigma_1 \Rightarrow \cdots \Rightarrow \sigma_n \Rightarrow \tau$ and $t_1 :: \sigma_1, \ldots, t_n :: \sigma_n$ then $\mathsf{f}(t_1, \ldots, t_n) :: \tau$

Varying the reduction relation

Reconsider: Toyama's example:

$$\begin{array}{rcl} \mathsf{f}(x,1,2) & \to & \mathsf{f}(x,x,x) \\ \mathsf{or}(x,y) & \to & x \\ \mathsf{or}(x,y) & \to & y \end{array}$$

Varying the reduction relation

Reconsider: Toyama's example:

$$\begin{array}{rcl} \mathsf{f}(x,1,2) & \to & \mathsf{f}(x,x,x) \\ \mathsf{or}(x,y) & \to & x \\ \mathsf{or}(x,y) & \to & y \end{array}$$

Non-terminating because: $\mathsf{f}(\mathsf{or}(1,2),1,2)$

Varying the reduction relation

Reconsider: Toyama's example:

$$\begin{array}{rcl} \mathsf{f}(x,1,2) & \to & \mathsf{f}(x,x,x) \\ \mathsf{or}(x,y) & \to & x \\ \mathsf{or}(x,y) & \to & y \end{array}$$

Non-terminating because: $\mathsf{f}(\underline{\mathsf{or}(1,2)},1,2)$

Definition (Rewrite Relation)

Let $\mathcal{R} = (\Sigma, R)$ be a TRS. The rewrite relation $\to_{\mathcal{R}}$ on $\mathcal{T}(\Sigma, \mathcal{X})$ is defined as:

$$C[\ell\sigma] \to_{\mathcal{R}} C[r\sigma]$$

for every:

- rule $\ell \to r \in R$,
- context C, and
- substitution σ.

Definition (Innermost Rewrite Relation)

Let $\mathcal{R} = (\Sigma, R)$ be a TRS. The innermost rewrite relation $\to_{\mathcal{R},in}$ on $\mathcal{T}(\Sigma, \mathcal{X})$ is defined as:

$$C[\ell\sigma] \to_{\mathcal{R}} C[r\sigma]$$

for every:

- rule $\ell \to r \in R$,
- context C, and
- substitution σ.

provided:

- every strict subterm of $\ell\sigma$ is in normal form with respect to $\to_{\mathcal{R}}$

Definition (Innermost Rewrite Relation)

Let $\mathcal{R} = (\Sigma, R)$ be a TRS. The innermost rewrite relation $\to_{\mathcal{R},in}$ on $\mathcal{T}(\Sigma, \mathcal{X})$ is defined as:

$$C[\ell\sigma] \to_{\mathcal{R}} C[r\sigma]$$

for every:

- rule $\ell \to r \in R$,
- context C, and
- substitution σ.

provided:

- every strict subterm of $\ell\sigma$ is in normal form with respect to $\to_{\mathcal{R}}$
- that is, if $t \lhd \ell\sigma$ then $t \neq \ell'\sigma'$ for any $\ell' \to r' \in R$

Definition (Rewrite Relation)

Let $\mathcal{R} = (\Sigma, R)$ be a TRS. The rewrite relation $\to_{\mathcal{R}}$ on $\mathcal{T}(\Sigma, \mathcal{X})$ is defined as:

$$C[\ell\sigma] \to_{\mathcal{R}} C[r\sigma]$$

for every:

- rule $\ell \to r \in R$,
- context C, and
- substitution σ.

Definition (Rewrite Relation)

Let $\mathcal{R} = (\Sigma, R)$ be a TRS. The rewrite relation $\to_{\mathcal{R}}$ on $\mathcal{T}(\Sigma, \mathcal{X})$ is defined as:

$$s[\ell\sigma]_p \to_{\mathcal{R}} s[r\sigma]_p$$

for every:

- rule $\ell \to r \in R$,
- term s and position p in $\mathcal{P}\mathrm{os}(s)$, and
- substitution σ.

Definition (Outermost Rewrite Relation)

Let $\mathcal{R} = (\Sigma, R)$ be a TRS. The outermost rewrite relation $\to_{\mathcal{R},out}$ on $\mathcal{T}(\Sigma, \mathcal{X})$ is defined as:

$$s[\ell\sigma]_p \to_{\mathcal{R}} s[r\sigma]_p$$

for every:

- rule $\ell \to r \in R$,
- term s and position p in $\mathcal{P}os(s)$, and
- substitution σ.

provided:

- for every $q < p$: $s|_q \neq \ell'\sigma'$ for any $\ell' \to r' \in R$