Automated Program Verification Environment APROVE

Presented by:

Mostafa Elsaie

10th of May, 2011

Outline

- Termination Provers
  - Aim?
- APROVE
  - Input Languages
  - Syntax
  - Proving Mechanism
  - GUI
  - Reading the Output

Termination Provers

- The aim is to find the answer for “Does program terminate?“
- Based on Termination *Proof*

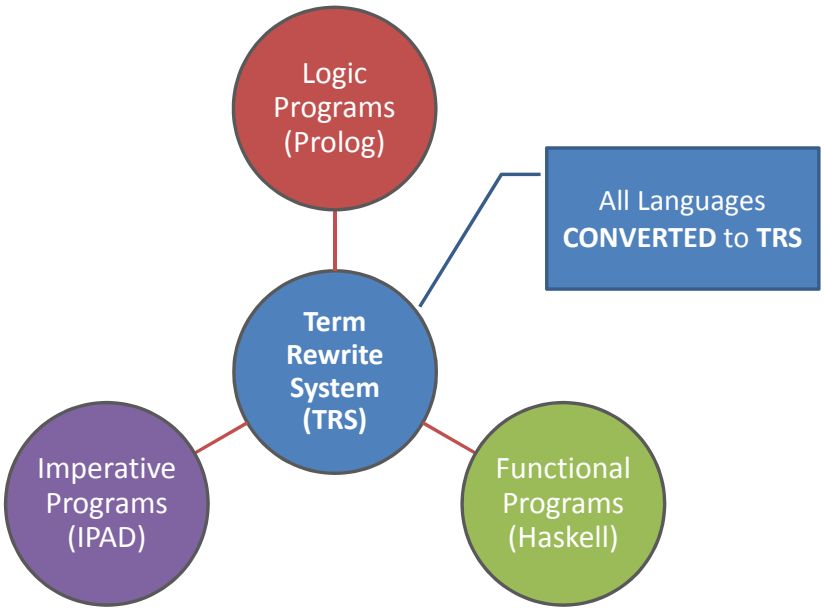
- *Example:*

```
i := 0
loop until i = SIZE_OF_DATA
   process_data(data[i]) //process the data chunk at position i
   i := i + 1 //move to the next chunk of data to be processed
```

APROVE - History

- Built at the “*RWTH Aachen University, Germany*”
- Developed by “*Research Group II of CS*”
- First Version released in 2001
- Most powerful prover for 04’,05’,06’…2010

APROVE - Input Languages



APROVE - TRS

- A TRS is a pair $(\sum, R)$
- The alphabet $\sum$ consists of:
  - Infinite set of variables *(x, y, z…)*
  - Non-empty set of Function symbols ***(F, G…)***
- The set of terms $Ter(\sum)$ is defined:
  - $(x, y, z..) \in Ter(\sum)$
  - $F(t_1, t_2, t_3...) \in Ter(\sum)$

APROVE - TRS

- A substitution α is:
  - Map from *Ter (∑)* to *Ter (∑)*
  - $\alpha(\ \mathbf{F}(t_1, t_2, t_3\ldots)\ ) == \mathbf{F}(\alpha(t_1), \alpha(t_2), \alpha(t_3)\ldots)$

- A rewrite rule *(A, B)*:
  - $A \rightarrow B$
  - LHS is not a variable
  - Variables in RHS already contained in LHS

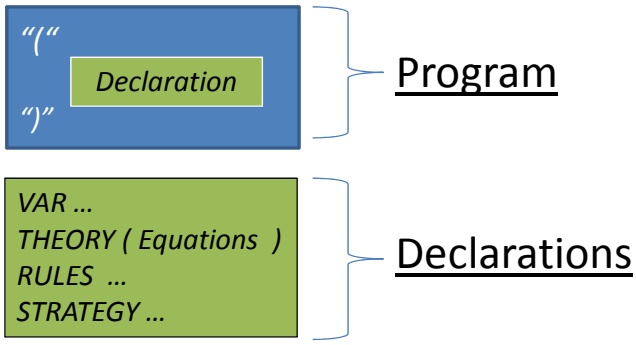
APROVE - TRS

- Example
  - Let $\sum = \{\mathbf{A}, \mathbf{M}, \mathbf{S}, 0\}$

r1: $\mathbf{A}(x,0)$	→	x
r2: $\mathbf{A}(x, \mathbf{S}(y))$	→	$\mathbf{S}(\mathbf{A}(x, y))$
r3: $\mathbf{M}(x, 0)$	→	0
r4: $\mathbf{M}(x, \mathbf{S}(y))$	→	$\mathbf{A}(\mathbf{M}(x, y), x)$
R5: $\mathbf{S}(x)$	→	$\mathbf{A}(x,1)$

- *Reduce* $\mathbf{M}(\mathbf{S}(\mathbf{S}(0)), \mathbf{S}(\mathbf{S}(0))) \rightarrow \mathbf{S}(\mathbf{S}(\mathbf{S}(\mathbf{S}(0))))$

APROVE - syntax



- Equations : *Term "==" Term.*
- RULES : *Term "->" Term.*
- Strategy : *INNERMOST*
  CONTEXTSENSITIVE (Variable int)

APROVE - syntax

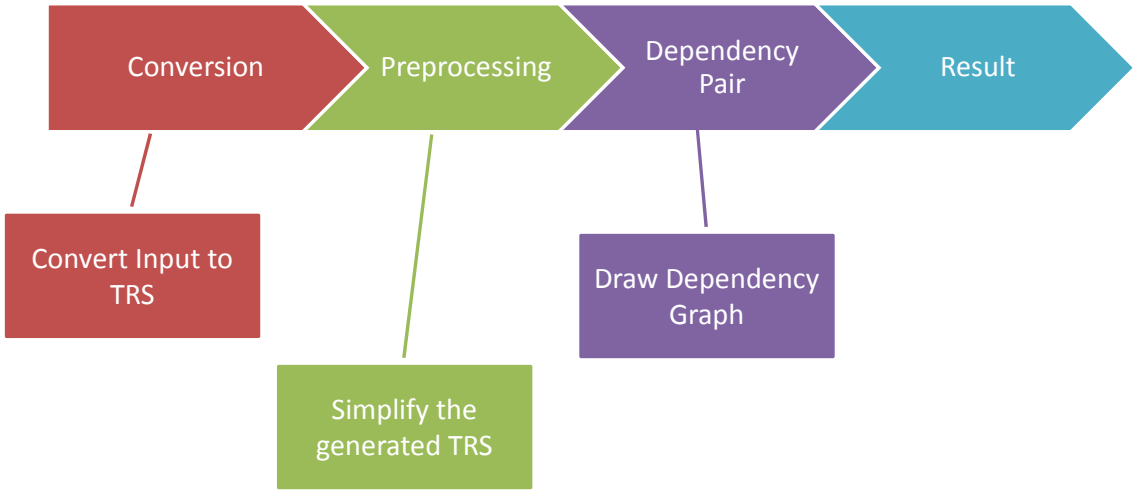
- Simple Example:

```
(VAR x y)
(RULES

  plus(s(s(x)),y) -> s(plus(x,s(y)))
  plus(x,s(s(y))) -> s(plus(s(x),y))
  plus(s(0),y) -> s(y)
  plus(0,y) -> y

  ack(0,y) -> s(y)
  ack(s(x),0) -> ack(x,s(0))
  ack(s(x),s(y)) -> ack(x,plus(y,ack(s(x),y)))
)
```

APROVE - proving mechanism



APROVE - preprocessing

- Aim?!
- Simplifying TRS for further processing

- Main techniques:
  - Removal of Redundant Rules (RRR)
  - Rule Reversal

APROVE - dependency pair

1. Construct List of Defined *"D"* symbols
   - Symbols present on the LHS
2. For every item in *"D"*, build list *"P"* containing all the rules *"A → B"* were *"B ∈ D"*
3. Repeat the same process on the output list

APROVE - dependency pair

- Example:

$$\begin{aligned}
&\mathrm{minus}(x, 0) \to x\\
&\mathrm{minus}(\mathrm{s}(x), \mathrm{s}(y)) \to \mathrm{minus}(x, y)\\
&\mathrm{quot}(0, \mathrm{s}(y)) \to 0\\
&\mathrm{quot}(\mathrm{s}(x), \mathrm{s}(y)) \to \mathrm{s}(\mathrm{quot}(\mathrm{minus}(x, y), \mathrm{s}(y)))
\end{aligned}$$

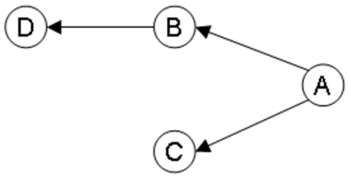
- Dependency pairs:

$$\begin{aligned}
&\mathrm{QUOT}(\mathrm{s}(x), \mathrm{s}(y)) \to \mathrm{QUOT}(\mathrm{minus}(x, y), \mathrm{s}(y))\\
&\mathrm{QUOT}(\mathrm{s}(x), \mathrm{s}(y)) \to \mathrm{MINUS}(x, y)\\
&\mathrm{MINUS}(\mathrm{s}(x), \mathrm{s}(y)) \to \mathrm{MINUS}(x, y)
\end{aligned}$$

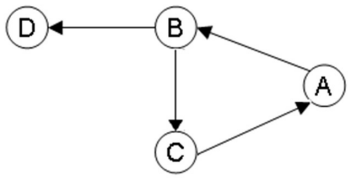
APROVE - dependency graph

- (A,B) models a dependency "a needs b evaluated first“

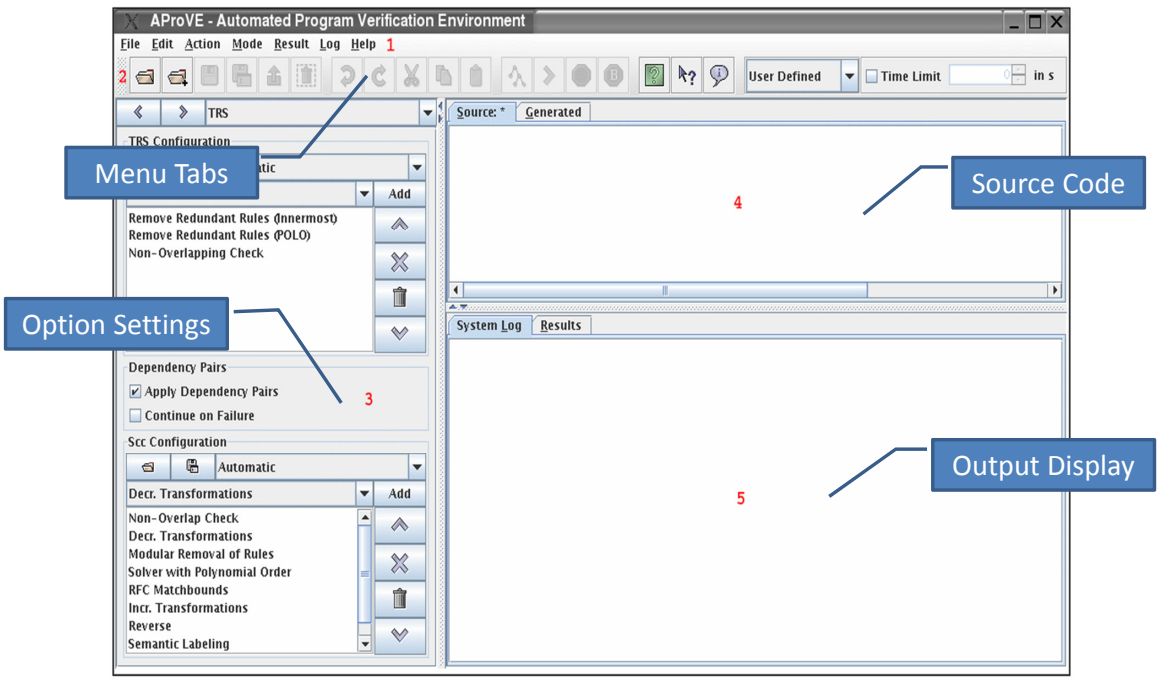
  - Given: A = B+C; B =5+D; C=4; D=2
  - Then R = (A,B),(A,C),(B,D)



  - Given: A = B+5; B =C*D; C=4A; D=2
  - Then R = (A,B),(B,C),(B,D),(C,A)



APROVE - GUI



APROVE - output

- The Output Consists of:-
  1) Proof Header:
     - *TRS Generated*
  2) Proof Body:
     - *Tree Like graph of techniques used*
  3) Proof Result:
     - *The Conclusion reached*

APROVE

Questions?