A Simple Framework for Program Verification and its Mechanical Certification in Theorema

Tudor Jebelean
Mădălina Erașcu

Research Institute for Symbolic Computation,
Johannes Kepler University, Linz, Austria

October 2012

Outline

Outline

The Theorema System

`www.theorema.org`

Aims at computer support for:

- constructing mathematical models (logical theories)
- defining and testing algorithms
- proving – for algorithm verification or synthesis

Natural style (human like):

- notation – formula appearance
- inference rules
- proof presentation

Outline

Syntax: Example

Program computing the GCD of two numbers using substractions

```
Program["G", G[a, b]],
 If[a = 0,
  Return[b]];
 If[b = 0,
  Return[a]];
 If[a ≥ b,
   a := G[a - b, b],
   a := G[a, b - a]]];
 Return[a],
Pre → a ≥ 0 ∧ b ≥ 0,
Post→ IsGCD[y, a, b]
```

Program: list of statements (assignment, conditional, return).

Contains also *terms* and *formulae* from the *object theory*.

Syntax

The meta-level predicate Π checks the syntactical correctness of the program:

- the program contains only valid constructs,
- variables are initialized,
- every program path contains a `Return`.

Definition

1. $\Pi[P] \Longleftrightarrow \Pi[\{\bar{x}\}, P]$
2. $\Pi[V, \langle \texttt{Return}[t] \rangle \frown P] \Longleftrightarrow \mathit{Vars}[t] \subseteq V$
3. $\Pi[V, \langle v := t \rangle \frown P] \Longleftrightarrow \bigwedge \left\{ \begin{array}{l} \mathit{Vars}[t] \subseteq V \\ \Pi[V \cup \{v\}, P] \end{array} \right.$
4. $\Pi[V, \langle \texttt{If}[\varphi, P_T, P_F] \rangle \frown P] \Longleftrightarrow \bigwedge \left\{ \begin{array}{l} \mathit{Vars}[\varphi] \subseteq V \\ \Pi[V, P_T \frown P] \\ \Pi[V, P_F \frown P] \end{array} \right.$

The definition of Π is a set of logical formulae!

Semantics: Example (1)

```
Program["G", G[a, b]],
 If[a = 0,
   Return[b]];
  If[b = 0,
   Return[a]];
  If[a ≥ b,
    a := G[a - b, b],
    a := G[a, b - a]]];
  Return[a],
 Pre → a ≥ 0 ∧ b ≥ 0,
 Post→ IsGCD[y, a, b]
```

Semantics

- $(a \geq 0) \wedge (b \geq 0) \wedge (a = 0) \implies G[a, b] = b$

Semantics: Example (2)

```
Program["G", G[a, b]],
  If[a = 0,
   Return[b]];
  If[b = 0,
   Return[a]];
  If[a ≥ b,
    a := G[a - b, b],
    a := G[a, b - a]]];
  Return[a],
 Pre → a ≥ 0 ∧ b ≥ 0,
 Post→ IsGCD[y, a, b]
```

Semantics

- $(a \geq 0) \wedge (b \geq 0) \wedge (a \neq 0) \wedge (b = 0) \implies G[a, b] = a$

Semantics: Example (3)

```
Program["G", G[a, b]],
  If[a = 0,
   Return[b]];
  If[b = 0,
   Return[a]];
  If[a ≥ b,
    a := G[a - b, b],
    a := G[a, b - a]]];
 Return[a],
 Pre → a ≥ 0 ∧ b ≥ 0,
 Post→ IsGCD[y, a, b]
```

Semantics

- $(a \geq 0) \wedge (b \geq 0) \wedge (a \neq 0) \wedge (b \neq 0) \wedge (a \geq b) \implies G[a, b] = G[a - b, b]$

Semantics: Example (4)

```
Program["G", G[a, b]],
  If[a = 0,
   Return[b]];
  If[b = 0,
   Return[a]];
  If[a ≥ b,
    a := G[a - b, b],
    a := G[a, b - a]]];
 Return[a],
 Pre → a ≥ 0 ∧ b ≥ 0,
 Post→ IsGCD[y, a, b]
```

Semantics

- $(a \geq 0) \wedge (b \geq 0) \wedge (a \neq 0) \wedge (b \neq 0) \wedge \neg(a \geq b) \implies G[a, b] = G[a, b - a]$

Semantics

The meta-level function Σ creates a formula with the shape:

$$\underset{\bar{x}:I_P}{\forall} \bigwedge \left\{ p_i[\bar{x}] \Rightarrow (f[\bar{x}] = g_i[\bar{x}]) \right\}_{i=1}^{n}$$

- This is a logical formula at object level.
- This is the implicit definition of the function implemented by the program.
- This is the functional program equivalent to the imperative one.

Definition

1. $\Sigma[P] = \underset{\bar{x}}{\forall}\big(I_P[\bar{x_0}] \;\Rightarrow\; \Sigma[\{\bar{x} \to \bar{x_0}\}, P]_{\{\bar{x}_0 \leftarrow \bar{x}\}}\big)$
2. $\Sigma[\sigma, \langle \texttt{Return}[t] \rangle \smile P] = (f[x_0] = t\sigma)$
3. $\Sigma[\sigma, \langle v := t \rangle \smile P] = \Sigma[\sigma \circ \{v \to t\sigma\}, P]$
4. $\Sigma[\sigma, \langle \texttt{If}[\varphi, P_T, P_F] \rangle \smile P] = \bigwedge \left\{ \begin{array}{l} \varphi\sigma \Longrightarrow \Sigma[\sigma, P_T \smile P] \\ \neg\varphi\sigma \Longrightarrow \Sigma[\sigma, P_F \smile P] \end{array} \right.$

Partial Correctness: Example (1)

```
Program["G", G[a, b]],
 If[a = 0,
  Return[b]];
  If[b = 0,
   Return[a]];
  If[a ≥ b,
    a := G[a - b, b],
    a := G[a, b - a]]];
  Return[a],
 Pre → a ≥ 0 ∧ b ≥ 0,
 Post→ IsGCD[y, a, b]
```

Verification Conditions

- $(a \geq 0) \wedge (b \geq 0) \wedge (a = 0) \implies IsGCD[b, a, b]$

Partial Correctness: Example (2)

```
Program["G", G[a, b]],
  If[a = 0,
   Return[b]];
  If[b = 0,
   Return[a]];
  If[a ≥ b,
    a := G[a - b, b],
    a := G[a, b - a]]];
  Return[a],
 Pre → a ≥ 0 ∧ b ≥ 0,
 Post→ IsGCD[y, a, b]
```

Verification Conditions

- $(a \geq 0) \wedge (b \geq 0) \wedge (a \neq 0) \wedge (b = 0) \implies IsGCD[a, a, b]$

Partial Correctness: Example (3)

```
Program["G", G[a, b]],
  If[a = 0,
   Return[b]];
  If[b = 0,
   Return[a]];
  If[a ≥ b,
    a := G[a - b, b],
    a := G[a, b - a]]];
 Return[a],
 Pre → a ≥ 0 ∧ b ≥ 0,
 Post→ IsGCD[y, a, b]
```

Verification Conditions

- $(a \geq 0) \wedge (b \geq 0) \wedge (a \neq 0) \wedge (b \neq 0) \wedge (a \geq b) \implies a - b \geq 0$
- $(a \geq 0) \wedge (b \geq 0) \wedge (a \neq 0) \wedge (b \neq 0) \wedge (a \geq b) \implies a - b \geq 0 \wedge b \geq 0$
- $(a \geq 0) \wedge (b \geq 0) \wedge (a \neq 0) \wedge (b \neq 0) \wedge (a \geq b) \wedge IsGCD[x, a - b, b] \implies IsGCD[x, a, b]$

Partial Correctness

The meta-level function Γ generates two kinds of verification conditions:

- coherence (safety) conditions;
- functional conditions.

Definition

1. $\Gamma[P] = \underset{\bar{x}}{\forall}\big(\Gamma[\{\bar{x} \to \bar{x}_0\},\ I_P[\bar{x}_0],\ P]_{\{\bar{x}_0 \leftarrow \bar{x}\}}\big)$
2. $\Gamma[\sigma,\ \Phi,\ \langle \mathtt{Return}[\gamma]\rangle \smile P] = \Big(\Phi \Rightarrow O_P[\bar{x}_0, \gamma\sigma]\Big)$
3. $\Gamma[\sigma,\ \Phi,\ \langle \mathtt{Return}[t_{\tau \leftarrow u[\overline{\gamma}]}]\rangle \smile P] =$
   $\Gamma[\sigma,\ \Phi,\ \langle w := u[\overline{\gamma}],\ \mathtt{Return}[t_{\tau \leftarrow w}]\rangle \smile P]$
4. $\Gamma[\sigma,\ \Phi,\ \langle v := \gamma\rangle \smile P] = \Gamma[\sigma \circ \{v \to \gamma\sigma\},\ \Phi,\ P]$

Partial Correctness

The meta-level function Γ generates two kinds of verification conditions:

- coherence (safety) conditions;
- functional conditions.

Definition

5. $\Gamma[\sigma,\ \Phi,\ \langle v := h[\overline{\gamma}]\rangle \smile P] = \bigwedge \left\{ \begin{array}{l} \Phi \Rightarrow I_h[\overline{\gamma}\sigma] \\ \Gamma[\sigma \circ \{v \to h[\overline{\gamma}\sigma]\},\ \Phi \wedge I_h[\overline{\gamma}\sigma],\ P] \end{array} \right.$
6. $\Gamma[\sigma,\ \Phi,\ \langle v := g[\overline{\gamma}]\rangle \smile P] = \bigwedge \left\{ \begin{array}{l} \Phi \Rightarrow I_g[\overline{\gamma}\sigma] \\ \Gamma[\sigma \circ \{v \to c\},\ \Phi \wedge I_g[\overline{\gamma}\sigma] \wedge O_g[\overline{\gamma}\sigma, c],\ P] \end{array} \right.$
7. $\Gamma[\sigma,\ \Phi,\ \langle v := t_{\tau \leftarrow u[\overline{\gamma}]}\rangle \smile P] = \Gamma[\sigma,\ \Phi,\ \langle w := u[\overline{\gamma}],\ v := t_{\tau \leftarrow w}\rangle \smile P]$
8. $\Gamma[\sigma,\ \Phi,\ \langle \mathtt{If}[\varphi_{\tau \leftarrow u[\overline{\gamma}]},\ P_T,\ P_F]\rangle \smile P] = \Gamma[\sigma,\ \Phi,\ \langle w := u[\overline{\gamma}],\ \mathtt{If}[\varphi_{\tau \leftarrow w},\ P_T,\ P_F]\rangle \smile P]$
9. $\Gamma[\sigma,\ \Phi,\ \langle \mathtt{If}[\varphi,\ P_T,\ P_F]\rangle \smile P] = \bigwedge \left\{ \begin{array}{l} \Gamma[\sigma,\ \Phi \wedge \varphi\sigma,\ P_T \smile P] \\ \Gamma[\sigma,\ \Phi \wedge \neg\varphi\sigma,\ P_F \smile P] \end{array} \right.$

Termination: Example

```
Program["G", G[a, b]],
 If[a = 0,
  Return[b]];
 If[b = 0,
  Return[a]];
 If[a ≥ b,
   a := G[a - b, b],
   a := G[a, b - a]]];
 Return[a],
Pre → a ≥ 0 ∧ b ≥ 0,
Post→ IsGCD[y, a, b]
```

Termination Condition

$$\left(\underset{\substack{a,b \\ a\geq 0, b\geq 0}}{\forall} \wedge \begin{cases} a = 0 \Rightarrow \pi[a,b] \\ b = 0 \Rightarrow \pi[a,b] \\ (a \neq 0 \wedge b \neq 0 \wedge a \geq b \wedge \pi[a-b,b]) \Rightarrow \pi[a,b] \\ (a \neq 0 \wedge b \neq 0 \wedge a \ngeq b \wedge \pi[a,b-a]) \Rightarrow \pi[a,b] \end{cases} \right) \Longrightarrow \left(\underset{\substack{a,b \\ a\geq 0, b\geq 0}}{\forall} \pi[a,b] \right)$$

The termination condition is expressed at object level!

Termination

The meta-level function Θ generates one termination condition.

Definition

1. $\Theta[P] = \left(\underset{\bar{x}:I_P}{\forall} \Theta[\{\bar{x} \to \bar{x_0}\},\ \mathbb{T},\ P]_{\{\bar{x}_0 \leftarrow \bar{x}\}}\right) \implies \underset{\bar{x}:I_P}{\forall} \pi[\bar{x}]$

2. $\Theta[\sigma,\ \Phi,\ \langle \texttt{Return}[\gamma]\rangle \frown P] = \left(\Phi \Rightarrow \pi[\bar{x_0}]\right)$

3. $\Theta[\sigma,\ \Phi,\ \langle v := \gamma\rangle \frown P] = \Theta[\sigma \circ \{v \to \gamma\sigma\},\ \Phi,\ P]$

4. $\Theta[\sigma,\ \Phi,\ \langle v := h[\overline{\gamma}]\rangle \frown P] = \Theta[\sigma \circ \{v \to h[\overline{\gamma}\sigma]\},\ \Phi,\ P]$

5. $\Theta[\sigma,\ \Phi,\ \langle v := f[\overline{\gamma}]\rangle \frown P] =$
$\Theta[\sigma \circ \{v \to y\},\ \Phi \wedge O_P[\overline{\gamma}\sigma, y] \wedge \pi[\overline{\gamma}\sigma],\ P]$

6. $\Theta[\sigma,\ \Phi,\ \langle v := g[\overline{\gamma}]\rangle \frown P] = \Theta[\sigma \circ \{v \to y\},\ \Phi \wedge O_g[\overline{\gamma}\sigma, y],\ P]$

7. $\Theta[\sigma,\ \Phi,\ \langle \texttt{If}[\varphi,\ P_T,\ P_F]\rangle \frown P] =$
$\bigwedge \begin{cases} \Theta[\sigma,\ \Phi \wedge \varphi\sigma,\ P_T \frown P] \\ \Theta[\sigma,\ \Phi \wedge \neg\varphi\sigma,\ P_F \frown P] \end{cases}$

Outline

Total Correctness

Generic Program

```
Program[``f'', f[x], If[Q[x], Return[S[x]],
Return[C[x,f[R[x]]]]
```

Semantics

$$\underset{x:I_f}{\forall} \wedge \begin{cases} Q[x] \Rightarrow (f[x] = S[x]) \\ \neg Q[x] \Rightarrow (f[x] = C[x, f[R[x]]]) \end{cases}$$

Partial correctness – functional conditions

$$\underset{x:I_f}{\forall} Q[x] \Rightarrow O_f[x, S[x]]$$

$$\underset{x:I_f}{\forall} \neg Q[x] \Rightarrow \underset{y}{\forall}(O_f[R[x], y] \Rightarrow O[x, C[x, y]])$$

Termination

$$\underset{x:I_f}{\forall} \wedge \left\{ \begin{cases} Q[x] \Rightarrow \pi[x] \\ \neg Q[x] \wedge \pi[R[x]] \Rightarrow \pi[x] \end{cases} \right\} \Rightarrow \underset{x:I_f}{\forall} \pi[x]$$

The proof of total correctness applies the induction principle (given by the termination condition) to the partial correctness conditions.
Take $\pi[x]$ as $O_f[x, f[x]]$.

Repetition function

Lemma

$$\forall_{h}\ \exists_{G}\ \forall_{x}\ \big(G[0,x] = x\ \wedge\ \forall_{n:\mathbb{N}}\big(G[n^{+},x] = h[G[n,x]]\big)\big)$$

Proof.

$$\forall_{m:\mathbb{N}}\ \exists_{H}\ \big(H[0] = x\ \wedge\ \forall_{n<m} H[n^{+}] = h[H[n]]\big) \quad \text{(natural induction on } m\text{)}$$

$$\exists_{\mathcal{H}}\ \forall_{m:\mathbb{N}}\ \big(\mathcal{H}[m][0] = x\ \wedge\ \forall_{n<m} \mathcal{H}[m][n^{+}] = h[\mathcal{H}[m][n]]\big) \quad \text{(Skolemize } H\text{)}$$

$$\forall_{n:\mathbb{N}}\ \forall_{m\geq n}\ \ H[m][n] = \mathcal{H}[n][n] \quad \text{(natural induction on } n\text{)}$$

$$\forall_{x}\ \exists_{g}\ \big(g[0] = x\ \wedge\ \forall_{n:\mathbb{N}} g[n^{+}] = h[g[n]]\big) \quad \text{(take } g[n] = \mathcal{H}[n][n]\text{)}$$

Skolemize g into $G[x]$ and use $G[n,x]$ instead of $G[x][n]$ □

$G[n,x]$ is usually denoted as $h^{n}[x]$.

Existence of the recursion index.

Lemma

$\underset{x:I_f}{\forall} \underset{n:\mathbb{N}}{\exists} (Q[R^n[x]] \wedge \underset{m<n}{\forall} \neg Q[R^m[x]])$.

(By the induction principle with appropriate $\pi[x]$)

Skolemize n to obtain $M[x]$ (the recursion index).

Existence of the implemented function

$$\exists_{f}\ \forall_{x:I_f}\ \wedge \begin{cases} Q[x] \Rightarrow (f[x] = S[x]) \\ \neg Q[x] \Rightarrow (f[x] = C[x, f[R[x]]]) \end{cases}$$

Proof. Similar to "Repetition function": use x (with $M[x]$) instead of n.

Natural induction on m:

$$\forall_{m:\mathbb{N}}\ \exists_{F}\ \forall_{x:I_f}\ (M[x] \leq m) \Rightarrow$$
$$\big((Q[x] \Rightarrow F[x] = S[x])\ \wedge (\neg Q[x] \Rightarrow F[x] = C[x, F[R[x]]])\big)$$

Skolemize F:

$$\exists_{\mathcal{F}}\ \forall_{m:\mathbb{N}}\ \forall_{x:I_f}\ (M[x] \leq m) \Rightarrow$$
$$\big((Q[x] \Rightarrow \mathcal{F}[m][x] = S[x])\ \wedge (\neg Q[x] \Rightarrow \mathcal{F}[m][x] = C[x, \mathcal{F}[m][R[x]]])\big)$$

Induction using $\pi[x]$:

$$\forall_{x:I_f}\ \forall_{m:\mathbb{N}} (m \geq M[x])\ \Rightarrow (\mathcal{F}[m][x] = \mathcal{F}[M[x]][n])$$

Finally: $f[x] = \mathcal{F}[M[x]][x]$.

Uniqueness: Take f_1, f_2 satisfying semantics and use induction with $\pi[x]$ as $f_1[x] = f_2[x]$.

Outline

Conclusion

Existence and Uniqueness

$$\left.\begin{array}{l}\text{object theory}\\ \text{semantics } \Sigma[P]\\ \text{verification conditions}\end{array}\right\} \models (\exists_{f}!\Sigma[P])$$

Mathematics: implicit function definition – needs existence [and uniqueness].

Programming: new function implementation – needs correctness.

These are equivalent!