

Formale Grundlagen der Informatik 3 –

8. Terminierungsbeweise

Christoph Walther
TU Darmstadt

1 Terminierende \mathcal{L} -Programme und Prozeduren

Definition 1 (Terminierung von \mathcal{L} -Programmen und Prozeduren)

- (1) Das initiale \mathcal{L} -Programm terminiert.¹
- (2) Ist P ein terminierendes \mathcal{L} -Programm und entsteht das \mathcal{L} -Programm P' durch Erweiterung von P durch eine Datentypdefinition, so terminiert P' .
- (3) Ist P ein terminierendes \mathcal{L} -Programm und entsteht das \mathcal{L} -Programm P' durch Erweiterung von P durch eine Lemmadefinition, so terminiert P' .
- (4) Ist P ein terminierendes \mathcal{L} -Programm und entsteht das \mathcal{L} -Programm P' durch Erweiterung von P durch die Definition einer Prozedur proc , so terminiert die Prozedur proc sowie das \mathcal{L} -Programm P' gdw. die *Rekursionsordnung*² $>_{R_{\text{proc}'}, x_1 \dots x_k}$ jeder monomorphen Instanz proc' von proc fundiert ist. ■

¹ Initiales \mathcal{L} -Programm = \mathcal{L} -Programm, das bei Start von *VeriFun* in *Predefined* angezeigt wird.

² Siehe Definition 6 in **Kapitel 7**.

Damit:

- Ein \mathcal{L} -Programm P *terminiert* gdw. jede \mathcal{L} -Prozedur von P terminiert.
- Rekursive Definition nach einer *fundierte* Relation garantiert *Terminierung*.
- Aus einer *terminierenden* Prozedur gewinnt man (uniform) eine *fundierte* Relation.

Satz 2 (Fundierte Relationenbeschreibungen und terminierenden Prozeduren)

Die Relationenbeschreibung R_p einer \mathcal{L} -Prozedur p ist fundiert gdw. die \mathcal{L} -Prozedur p terminiert.

Beweis: R_p ist fundiert gdw. jede monomorphe Instanz $R_{p'}$ von R_p fundiert ist (\Rightarrow **Kapitel 6, Definition 8**) gdw. jede Relation $>_{R_{p'}, x_1 \dots x_k}$ fundiert ist (\Rightarrow **Kapitel 6, Definition 8**) gdw. gdw. p terminiert (\Rightarrow **Definition 1**).

Satz 3

Für jedes terminierende \mathcal{L} -Programm ist $(\mathcal{G}(P), \Rightarrow_P)$ eine fundierte Menge.³

³ Siehe. Abschnitt 2.2 in **Kapitel 5**.

VeriFun:

- In *VeriFun* können nur *terminierende* \mathcal{L} -Programme *verifiziert* werden, also keine Beweise *partieller* sondern nur *totaler* Korrektheit !
- **Konsequenz:** Die *Terminierung* von Prozeduren muß *bewiesen* werden, um deren Eigenschaften beweisen zu können.

2 Automatische Terminierungsbeweise

- In *VeriFun* ist ein Verfahren zum automatischen Terminierungsnachweis implementiert.
- Bei Eingabe einer Prozedur p erzeugt das System eine Menge von *Terminierungshypothesen* th_1, \dots, th_n für p .
- Terminierungshypothesen haben die gleiche *Form wie Lemmata*.
- Das System versucht alle *Terminierungshypothesen* zu *verifizieren*.
- **Es gilt:** Sind alle *Terminierungshypothesen* th_1, \dots, th_n *wahr*, so *terminiert* die *Prozedur* p .

- **Mögliche Ergebnisse** bei Eingabe einer Prozedur p:
 - Die Erzeugung von Terminierungshypothesen scheitert
(*Prozedur im Zustand ignored*; graues *Prozedur-Icon*)
 - ⇒ Benutzer muß durch Angabe von *Maßtermen* die Erzeugung geeigneter Terminierungshypothesen initiieren (s. Abschnitt 3).
 - Die Erzeugung von Terminierungshypothesen gelingt, System scheitert jedoch beim Nachweis derselben
(*Prozedur im Zustand ready*; blaues *Prozedur-Icon*)
 - ⇒ Benutzer muß interaktiv den Beweisbaum der Terminierungshypothesen editieren (=> *HPL-Regeln*) *oder aber* durch Angabe von *Maßtermen* die Erzeugung alternativer Terminierungshypothesen initiieren (s. Abschnitt 3).
 - Die Erzeugung der Terminierungshypothesen sowie deren Beweis gelingt
 - ⇒ **fertig !** Terminierung von p bewiesen
(*Prozedur im Zustand verified*; grünes *Prozedur-Icon*).

- Das automatische Terminierungsbeweisverfahren ist *korrekt* aber *unvollständig*, d.h. es scheitert bei gewissen *terminierenden* Prozeduren:
 - Deshalb sind auch *interaktive* Terminierungsbeweise erforderlich.
 - Diese sind auch *prinzipiell* erforderlich, denn es ist *unmöglich* vollständige (und korrekte) automatische Terminierungsbeweisverfahren anzugeben (=> Vorlesung *Berechenbarkeit*: Nicht-berechenbare Funktionen).
- **Aber:** Das *automatische Terminierungsbeweisverfahren* ist für alle Prozeduren *erfolgreich*, deren *Rekursionsordnung* einer *strukturellen Ordnung* entsprechen (=> Definition 3 in **Kapitel 7**).
 - *Anmerkung:* Dies ist sogar *semi-entscheidbar* (=> Vorlesung *Berechenbarkeit*: Primitiv-rekursive Funktionen).
- **Beispiel:** Alle Prozeduren aus der *InsertionSort*-Fallstudie (=> **Kapitel 2**)

3 Interaktive Terminierungsbeweise

- “Meta”-Axiom in *VeriFun*:

(*) Prozedur $>$ (aus *Predefined*) definiert eine *fundierte Relation* auf der Menge der Konstruktorgrundterme von nat .

- **Sonderfall:** Wir betrachten zunächst nur Prozeduren p mit *genau einem formalen Parameter* x von Typ nat und *genau einem rekursiven Aufruf*.
- Sei $A = \langle H, \{\{x/t\}\} \rangle$ die atomare rekursive Relationenbeschreibung von p .
- Aus A bilden wir die *Terminierungshypothese*

$$th_A = \forall x:\text{nat} \text{ if}\{\text{AND}(H), x > t, \text{true}\}$$

wobei $\text{AND}(\{b_1, \dots, b_n\}) =$ Konjunktion der b_i ausgedrückt mittels *if*-Termen.

Satz 4

Kann die Terminierungshypothese th_A verifiziert werden, so terminiert p .

Beweis:

- Angenommen, p terminiert nicht.
- Dann gibt es eine unendliche Folge q_0, q_1, q_2, \dots von Konstruktorgrundtermen vom Typ nat , so daß für alle $i \in \mathbb{N}$ gilt:
Bei Berechnung von $p(q_i)$ wird $p(q_{i+1})$ rekursiv aufgerufen.
- Damit gilt $\text{eval}_P(\text{AND}(H[\mathbf{x}/q_i])) = \text{true}$ für alle $i \in \mathbb{N}$, denn die Bedingung $\text{AND}(H[\mathbf{x}/q_i])$, die zum jeweils nächsten rekursiven Aufruf führt, muß ja gelten.
- Damit gilt $\text{eval}_P(q_i > t[\mathbf{x}/q_i]) = \text{true}$ für alle $i \in \mathbb{N}$, denn th_A ist nach Voraussetzung verifiziert.
- Folglich gilt $q_i > \text{eval}_P(t[\mathbf{x}/q_i])$ für alle $i \in \mathbb{N}$.
- Für $r_0 := q_0$ und $r_{i+1} := \text{eval}_P(t[\mathbf{x}/r_i])$ für alle $i \in \mathbb{N}$ ist r_0, r_1, r_2, \dots eine unendliche Folge von Konstruktorgrundtermen vom Typ nat mit $r_i > r_{i+1}$.
- Damit ist $>$ im Widerspruch zu (*) nicht fundiert.
- Folglich war die Annahme falsch, d.h. p terminiert. ■

Beispiel 1

Für

```
function half(x:nat):nat <=
  if ?0(x)
    then 0
    else if ?0(^(x)) then 0 else +(half(^(^(x)))) end_if
end_if
```

erhält man die rekursive atomare Relationenbeschreibung

$$A = \langle \{ \neg ?0(x), \neg ?0(^{(x)}) \}, \{ \{ x / ^{(x)} \} \} \rangle$$

und damit die (trivial zu verifizierende) Terminierungshypothese^{4,5}

$$th_A = \forall x:nat \text{ if } \{ \neg ?0(x), \text{ if } \{ \neg ?0(^{(x)}), x > ^{(x)}, \text{ true} \}, \text{ true} \} . \quad (1)$$

⁴ Nach Definition müssten wir eigentlich $\forall x:nat \text{ if } \{ \text{if } \{ \neg ?0(x), \neg ?0(^{(x)}), \text{ false} \}, x > ^{(x)}, \text{ true} \}$ schreiben. Wir verwenden jedoch in den Beispielen die *normalisierte* Form (1) (d.h. $a \rightarrow (b \rightarrow c)$ anstatt $a \wedge b \rightarrow c$), die auch in *VeriFun* angezeigt wird.

⁵ Eine Terminierungshypothese für die Prozedur `half` wird in *VeriFun* automatisch erzeugt (und bewiesen).

- **Verallgemeinerung 1:** Prozedur p enthält *mehrere* rekursive Aufrufe
- *Dann:*
 - Bilde für *jede* atomare rekursive Relationenbeschreibung

$$A = \langle H, \{\{x/t_1\}, \dots, \{x/t_j\}\} \rangle$$

die Terminierungshypothesen

$$th_{A,i} = \forall x:\text{nat} \text{ if } \{\text{AND}(H), x > t_i, \text{true}\} .$$

- Verifiziere alle Terminierungshypothesen $th_{A,1}, \dots, th_{A,j}$.

Beispiel 2

Für die Prozedur zur Berechnung der Fibonacci-Zahlen

```
function fib(x:nat):nat <=
  if ?0(x)
    then 0
    else if ?0(^(x)) then 1 else fib(^(x))+fib(^(^(x))) end_if
end_if
```

erhält man die rekursive atomare Relationenbeschreibung

$$A = \langle \{ \neg ?0(x), \neg ?0(^{(x)}) \}, \{ \{ x / ^{(x)} \}, \{ x / ^{(^{(x)})} \} \} \rangle$$

und damit die (trivial zu verifizierenden) Terminierungshypothesen⁶

$$th_{A,1} = \forall x:nat \text{ if } \{ \neg ?0(x), \text{ if } \{ \neg ?0(^{(x)}), x > ^{(x)}, \text{ true} \}, \text{ true} \}$$

sowie

$$th_{A,2} = \forall x:nat \text{ if } \{ \neg ?0(x), \text{ if } \{ \neg ?0(^{(x)}), x > ^{(^{(x)})}, \text{ true} \}, \text{ true} \}.$$

⁶ Terminierungshypothesen für die Prozedur fib werden in *VeriFun* automatisch erzeugt (und bewiesen).

- **Verallgemeinerung 2:**

Der formale Parameter x der Prozedur p ist vom Typ $\tau \neq \text{nat}$

- *Dann:*

- Definiere eine *Terminierungsfunktion* $m : \tau \rightarrow \text{nat}$ durch Angabe eines *Maßterms* m vom Typ nat , der x als einzige Variable enthält.
- Bilde für *jede* atomare rekursive Relationenbeschreibung

$$A = \langle H, \{\{x/t\}\} \rangle$$

die Terminierungshypothese

$$th_A = \forall x:\tau \text{ if } \{\text{AND}(H), m > m[x/t], \text{true}\} .$$

- Verifiziere die Terminierungshypothese th_A .

Beispiel 3

Für die Prozedur

```
function minsort(k:list[nat]):list[nat] <=
  if ?∅(k)
  then ∅
  else minimum(k) :: minsort(k \ minimum(k))
end_if
```

definieren wir eine Prozedur (zur Berechnung der Länge einer polymorphen Liste)

```
function [outfix] |(l:list[@T]):nat <=
  if ?∅(l) then 0 else +( | tl(l) | ) end_if
```

Für minsort erhält man die rekursive atomare Relationenbeschreibung

$$A = \langle \{ \neg ?\emptyset(k) \}, \{ \{ k/k \setminus \text{minimum}(k) \} \} \rangle$$

Als Maßterm wählen wir $|k|$ und erhalten die Terminierungshypothese

$$th_A = \forall k:\text{list}[\text{nat}] \text{ if } \{ \neg ?\emptyset(k), |k| > |k \setminus \text{minimum}(k)|, \text{true} \}.$$

Bemerkung 1

- (1) In Beispiel 3 wird die *Terminierung* von $|\dots|$ *vorausgesetzt*; dies ist erlaubt, denn der *Terminierungsnachweis* von $|\dots|$ gelingt *automatisch* (wg. struktureller Rekursion).
- (2) Beim *automatische Terminierungsnachweis* von `minsort` erzeugt *VeriFun* eine wahre Terminierungshypothese, scheitert jedoch bei deren Beweis; *Grund*: Es fehlt ein Lemma zur Vervollständigung des Beweises (\Rightarrow Übung).
- (3) Genaugenommen haben wir schon in den Beispielen 1 und 2 Maßterme verwendet (\Rightarrow *Überlegen*: Wie lauten die Maßterme dort?).

- **Verallgemeinerung 3a:** Die Prozedur p besitzt $n \geq 1$ formale Parameter $\mathbf{x}_1, \dots, \mathbf{x}_n$ der Typen τ_1, \dots, τ_n
- *Dann:*
 - Definiere eine *Terminierungsfunktion* $m : \tau_1 \times \dots \times \tau_n \rightarrow \text{nat}$ durch Angabe eines *Maßterms* m vom Typ nat , der nur $\mathbf{x}_1, \dots, \mathbf{x}_n$ als Variable enthält.
 - Bilde für *jede* atomare rekursive Relationenbeschreibung

$$A = \langle H, \{\{x_1/t_1, \dots, x_n/t_n\}\} \rangle$$

die Terminierungshypothese $th_A =$

$$\forall \mathbf{x}_1 : \tau_1, \dots, \mathbf{x}_n : \tau_n \text{ if } \{\text{AND}(H), m > m[\mathbf{x}_1/\mathbf{t}_1, \dots, \mathbf{x}_n/\mathbf{t}_n], \text{true}\}.$$

- Verifiziere jede Terminierungshypothese th_A .

Beispiel 4*Für die Prozedur*

```

function gcd(x:nat, y:nat) : nat <=
if ?0(x)
  then y
  else if ?0(y)
    then x
    else if x > y then gcd(x - y, y) else gcd(x, y - x) end_if
  end_if
end_if

```

erhält man die rekursiven atomare Relationenbeschreibungen

$$A_1 = \langle \{ \neg ?0(x), \neg ?0(y), x > y \}, \{ \{ x/x - y, y/y \} \} \rangle$$

sowie

$$A_2 = \langle \{ \neg ?0(x), \neg ?0(y), \neg x > y \}, \{ \{ x/x, y/y - x \} \} \rangle .$$

Als Maßterm wählen wir $x + y$ und erhalten die Terminierungshypothesen

$$th_{A_1} = \forall x, y : \text{nat}$$

$$\text{if}\{\neg?0(x), \text{if}\{\neg?0(y), \text{if}\{x > y, x + y > (x - y) + y, \text{true}\}, \dots\}$$

$$\text{sowie } th_{A_2} = \forall x, y : \text{nat}$$

$$\text{if}\{\neg?0(x), \text{if}\{\neg?0(y), \text{if}\{\neg x > y, x + y > x + (y - x), \text{true}\}, \dots\}.$$

Bemerkung 2

- (1) In Beispiel 4 wird die *Terminierung* der Prozedur $+$ vorausgesetzt; dies ist erlaubt, denn der *Terminierungsnachweis* von $+$ gelingt *automatisch* (wg. struktureller Rekursion).
- (2) Der *Terminierungsnachweis* von gcd gelingt ebenfalls *automatisch* (\Rightarrow ausprobieren !); interaktiv hier nur zur Illustration.
- (3) Ein Maßterm enthält *mindestens einen* und *höchstens alle* formalen Parameter einer Prozedur; jedoch *müssen nicht alle* formalen Parameter im Maßterm vorkommen. (\Rightarrow Überlegen: Kann die Terminierung von gcd mittels eines Maßterms gezeigt werden, in dem *nur* x vorkommt ?)

Beispiel 5 (*Nicht alle formalen Parameter im Maßterm*)*Für die Prozedur*

```

function find.max.up( a:list[nat], i:nat, x:nat ):nat <=
  if | a | > i
    then let a.i := (a !! i) in
      if a.i > x
        then find.max.up( a, +(i), a.i )
        else find.max.up( a, +(i), x )
      end_if
    end_let
  else x
end_if

```

erhält man die rekursiven atomare Relationenbeschreibungen

$$A_1 = \langle \{ |a| > i, (a !! i) > x \}, \{ \{ a/a, i/+(i), x/(a !! i) \} \} \rangle$$

sowie

$$A_2 = \langle \{ |a| > i, \neg(a !! i) > x \}, \{ \{ a/a, i/+(i), x/x \} \} \rangle .$$

Als Maßterm wählen wir $|a| - i$ (und ignorieren damit den formalen Parameter x). Damit erhält man die Terminierungshypothesen

$$th_{A_1} = \forall a:\text{list}[\text{nat}], i:\text{nat}, x:\text{nat}$$

$$\text{if}\{|a| > i, \text{if}\{(a !! i) > x, |a| - i > |a| - ^+(i), \text{true}\}, \text{true}\}$$

$$\text{sowie } th_{A_2} = \forall a:\text{list}[\text{nat}], i:\text{nat}, x:\text{nat}$$

$$\text{if}\{|a| > i, \text{if}\{\neg(a !! i) > x, |a| - i > |a| - ^+(i), \text{true}\}, \text{true}\}.$$

Bemerkung 3

- (1) In Beispiel 5 wird die *Terminierung* von $|\dots|$ vorausgesetzt; dies ist erlaubt, denn der *Terminierungsnachweis* von $|\dots|$ gelingt *automatisch* (wg. struktureller Rekursion).
- (2) Der *automatische Terminierungsnachweis* von `find.max.up` scheitert bei Erzeugung gültiger Terminierungshypothesen, d.h. alle erzeugten Terminierungshypothesen sind *falsch* (und somit *nicht beweisbar*).

- **Verallgemeinerung 3b:** Die Prozedur p besitzt $n \geq 1$ formale Parameter $\mathbf{x}_1, \dots, \mathbf{x}_n$ der Typen τ_1, \dots, τ_n
- *Dann:*
 - Definiere *Terminierungsfunktionen* $m_1, \dots, m_j : \tau_1 \times \dots \times \tau_n \rightarrow \mathbf{nat}$ durch Angabe einer *Liste von Maßtermen* $\mathbf{m}_1, \dots, \mathbf{m}_j$ jeweils vom Typ \mathbf{nat} , die nur $\mathbf{x}_1, \dots, \mathbf{x}_n$ als Variable enthalten.
 - Bilde für *jede* atomare rekursive Relationenbeschreibung

$$A = \langle H, \{ \{ x_1/t_1, \dots, x_n/t_n \} \} \rangle$$

eine Terminierungshypothese entsprechend der *links-lexikographischen Kombination* der Maßterme. Für $j = 2$ erhält man z.B. $th_A =$

$$\begin{aligned} \forall \mathbf{x}_1 : \tau_1, \dots, \mathbf{x}_n : \tau_n \text{ if } \{ & \text{AND}(H), \\ & \text{if } \{ \mathbf{m}_1 > \mathbf{m}_1 [\mathbf{x}_1/\mathbf{t}_1, \dots, \mathbf{x}_n/\mathbf{t}_n], \\ & \text{true}, \\ & \text{if } \{ \mathbf{m}_1 = \mathbf{m}_1 [\mathbf{x}_1/\mathbf{t}_1, \dots, \mathbf{x}_n/\mathbf{t}_n], \\ & \quad \mathbf{m}_2 > \mathbf{m}_2 [\mathbf{x}_1/\mathbf{t}_1, \dots, \mathbf{x}_n/\mathbf{t}_n], \\ & \quad \text{false} \} \}, \\ & \text{true} \}. \end{aligned}$$

- Verifiziere jede Terminierungshypothese th_A .

Beispiel 6 (*Lexikographische Kombination von Maßtermen*)

Für die Prozedur `gcd` wählen wir x, y als Maßtermliste und erhalten damit die Terminierungshypothesen

$th_{A_1} =$

$$\begin{aligned} \forall x, y : \text{nat} \text{ if} \{ \neg ?0(x), \\ \quad \text{if} \{ \neg ?0(y) \}, \\ \quad \quad \text{if} \{ x > y, \\ \quad \quad \quad \text{if} \{ x > x - y, \text{true}, \text{if} \{ x = x - y, y > y, \text{false} \} \}, \\ \quad \quad \quad \text{true} \}, \\ \quad \quad \text{true} \}, \\ \quad \text{true} \} \end{aligned}$$

sowie $th_{A_2} =$

$$\begin{aligned} \forall x, y : \text{nat} \text{ if} \{ \neg ?0(x), \\ \quad \text{if} \{ \neg ?0(y) \}, \\ \quad \quad \text{if} \{ \neg x > y, \\ \quad \quad \quad \text{if} \{ x > x, \text{true}, \text{if} \{ x = x, y > y - x, \text{false} \} \}, \\ \quad \quad \quad \text{true} \}, \\ \quad \quad \text{true} \}, \\ \quad \text{true} \}. \end{aligned}$$

Beispiel 7 (*Lexikographische Kombination von Maßtermen*)*Für die Prozedur*

```
function flatten(x:sexpr[@ITEM]) :sexpr[@ITEM] <=
case x of
  cons : case car(x) of
    cons : flatten(cons(car(car(x)),
                        cons(cdr(car(x)), cdr(x)))),
                  cons(car(x), flatten(cdr(x))))
    other : cons(car(x), flatten(cdr(x)))
  end_case,
  other : x
end_case
```

(zur Linearisierung eines Binärbaums) erhält man die rekursiven atomaren Relationenbeschreibungen

$$A_1 = \langle \{ ?cons(x), ?cons(car(x)) \}, \{ \{ x / cons(car(car(x)), cons(cdr(car(x)), cdr(x))) \} \} \rangle$$

sowie

$$A_2 = \langle \{ ?cons(x), \neg ?cons(car(x)) \}, \{ \{ x / cdr(x) \} \} \rangle .$$

Als Maßtermliste wählen wir $\#nodes(x)$, $\#nodes(car(x))$ (\Rightarrow **Kapitel 5**, Abschnitt 1.3.1) und erhalten damit die Terminierungshypothesen

$th_{A_1} =$

$$\forall x : \text{sexpr} \text{ if} \{ ?\text{cons}(x),$$

$$\quad \text{if} \{ ?\text{cons}(\text{car}(x)),$$

$$\quad \quad \text{if} \{ \#nodes(x) > \#nodes(\text{cons}(\dots)),$$

$$\quad \quad \quad \text{true},$$

$$\quad \quad \quad \text{if} \{ \#nodes(x) = \#nodes(\text{cons}(\dots)),$$

$$\quad \quad \quad \quad \#nodes(\text{car}(x)) > \#nodes(\text{car}(\text{cons}(\dots))),$$

$$\quad \quad \quad \quad \text{false} \},$$

$$\quad \quad \text{true} \},$$

$$\text{true} \}$$

sowie

$$th_{A_2} =$$

$$\begin{aligned} \forall x : \text{sexpr} \text{ if} \{ & ?\text{cons}(x), \\ & \text{if} \{ \neg ?\text{cons}(\text{car}(x)), \\ & \text{if} \{ \#nodes(x) > \#nodes(\text{cdr}(x)), \\ & \text{true}, \\ & \text{if} \{ \#nodes(x) = \#nodes(\text{cdr}(x)), \\ & \#nodes(\text{car}(x)) > \#nodes(\text{car}(\text{cdr}(x))), \\ & \text{false} \} \}, \\ & \text{true} \}, \\ & \text{true} \}. \end{aligned}$$

Bemerkung 4

- (1) In Beispiel 5 wird die *Terminierung* der Prozedur `#nodes` (\Rightarrow **Kapitel 5**, Abschnitt 1.3.1) *vorausgesetzt*; dies ist erlaubt, denn der *Terminierungsnachweis* von `#nodes` gelingt *automatisch* (wg. struktureller Rekursion).
- (2) Der *automatische Terminierungsnachweis* von `flatten` scheitert, da *VeriFun* keine Terminierungshypothesen erzeugen kann.

Mehrere Terminierungsbeweise (für *eine* Prozedur)

- Eine Prozedur p kann bezüglich *mehrerer* (unterschiedlicher) fundierten Relationen $\mathcal{R}_1, \dots, \mathcal{R}_h$ terminieren.
- Die Terminierung von p ist gezeigt, sobald **alle** Terminierungshypothesen, die mittels **eines** Maßterms (**einer** Maßtermliste) erzeugt wurden, bewiesen worden sind !
- **Aber:**
 - Für jeden Terminierungsnachweis bzgl. *eines* Maßterms (*einer* Maßtermliste) erhält man *eine* fundierte Relationenbeschreibung.
 - Aus *fundierten* Relationenbeschreibungen werden (*gültige*) Induktionsaxiome erzeugt (\Rightarrow **Kapitel 6**, Abschnitt 2.3).
 - *Damit:* Für jeden Terminierungsnachweis bzgl. *eines* Maßterms (*einer* Maßtermliste) erhält man *ein* Induktionsaxiom.
 - *Konsequenz:* Es ist vorteilhaft – falls möglich – mehrere Terminierungsbeweise für eine Prozedur zu führen, da so mehrere (unterschiedliche) Induktionsaxiome für nachfolgende Beweisaufgaben zur Verfügung stehen.

Beispiel 8

Für die Prozedur

```
function ntl(n : ℕ, k : list[@ITEM]) : list[@ITEM] <=
if ?∅(k)
then k
else if ?0(n)
      then k
      else ntl(^(n), tl(k))
end_if
end_if
```

erhält man die rekursive atomare Relationenbeschreibung

$$A = \langle \{ \neg ?\emptyset(k), \neg ?0(n) \}, \{ \{ n / ^{(n)}, k / tl(k) \} \} \rangle .$$

Als Maßterme wählen wir (1.) n und (2.) $|k|$ und erhalten damit die Terminierungshypothesen $th_{A,1} = \forall k : list[@ITEM], n : nat$

$$if \{ \neg ?\emptyset(k), if \{ \neg ?0(n), n > ^{(n)}, true \}, true \}$$

sowie $th_{A,2} = \forall k : list[@ITEM], n : nat$

$if\{\neg?0(k), if\{\neg?0(n), |k| > |tl(k)|, true\}, true\}.$

- Bei Beweis von $th_{A,1}$ wird die Bedingung “ $\neg?0(k)$ ” nicht verwendet. Damit erhält man die *optimierte Relationenbeschreibung* (\Rightarrow **Kapitel 7**, Abschnitt 3)

$$R_{nt1}^{(1)} := \left\{ \begin{array}{l} \langle \{?0(n)\}, \emptyset \rangle, \\ \langle \{\neg?0(n)\}, \{\{n/pred(n)\}\} \rangle \end{array} \right\}.$$

- Bei Beweis von $th_{A,2}$ wird die Bedingung “ $\neg?0(n)$ ” nicht verwendet. Damit erhält man die *optimierte Relationenbeschreibung* (\Rightarrow **Kapitel 7**, Abschnitt 3)

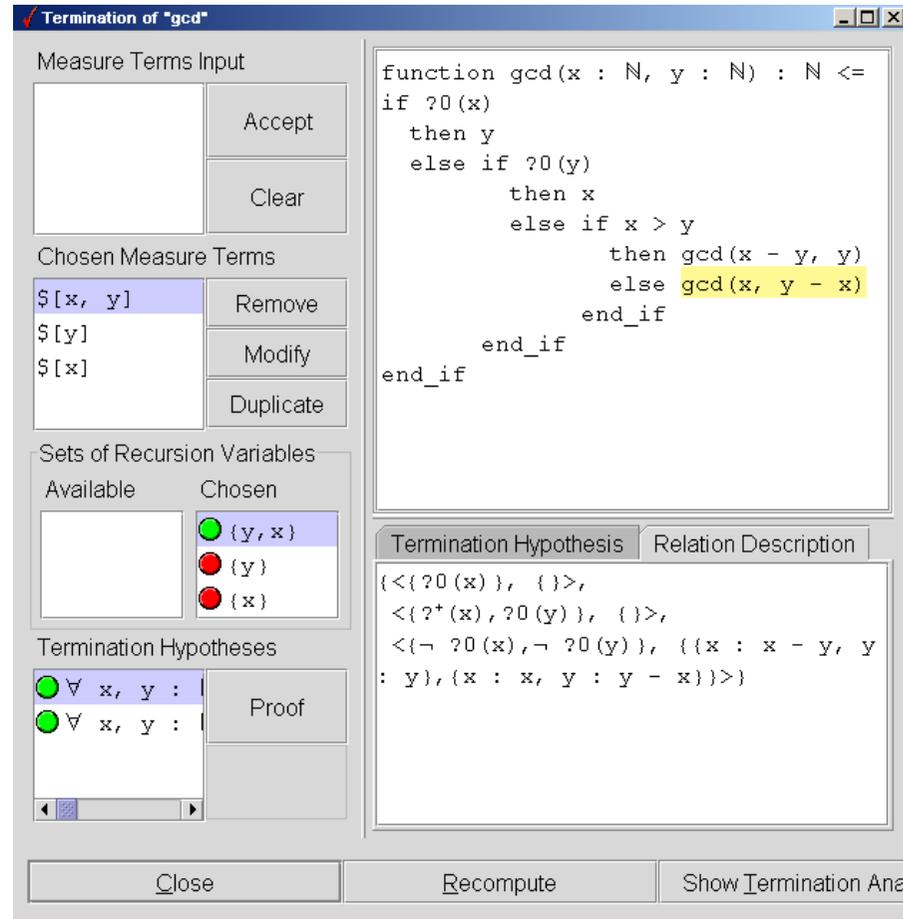
$$R_{nt1}^{(2)} := \left\{ \begin{array}{l} \langle \{?0(k)\}, \emptyset \rangle, \\ \langle \{\neg?0(k)\}, \{\{k/tl(k)\}\} \rangle \end{array} \right\}.$$

Bemerkung 5

- (1) Anstatt Maßterme können auch Listen von Maßtermen der Reihe nach angegeben werden.
- (2) Beide *Terminierungsnachweise* von $nt1$ gelingen *automatisch* (\Rightarrow ausprobieren !); interaktiv hier nur zur Illustration.

4 Benutzerinteraktion bei Terminierungsbeweisen

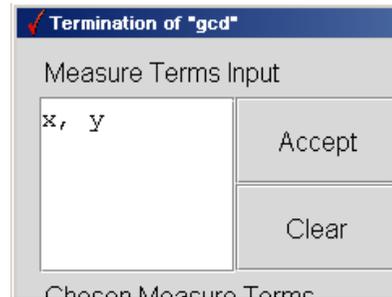
- Interaktive Terminierungsbeweise werden in *VeriFun* über das *Termination Window* analysiert und gesteuert
- **Aufruf:**
 1. Selektiere Prozedur-Icon im *Programm Window*
 2. Öffnen des *Termination Window* über
 - * Maus Doppelklick = Menue *Program\Set Termination* (falls Prozedur noch nicht in Beweisen verwendet wurde) oder
 - * *Termination Details* unter dem Reiter *Termination* im *Program Viewer*



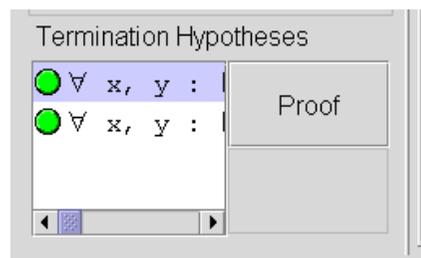
- Versuch automatischer Terminierungsbeweis mit Maßterm x
 - gescheitert, da (eine) Terminierungshypothese widerlegt werden kann
- Versuch automatischer Terminierungsbeweis mit Maßterm y
 - gescheitert, da (eine) Terminierungshypothese widerlegt werden kann
- Versuch automatischer Terminierungsbeweis mit Maßterm $x + y$
 - erfolgreich, da beide Terminierungshypothese bewiesen werden können

Interaktiver Terminierungsbeweis

- Nach Eingabe eines Maßterms (oder einer Liste von Maßtermen) *Accept* “drücken”



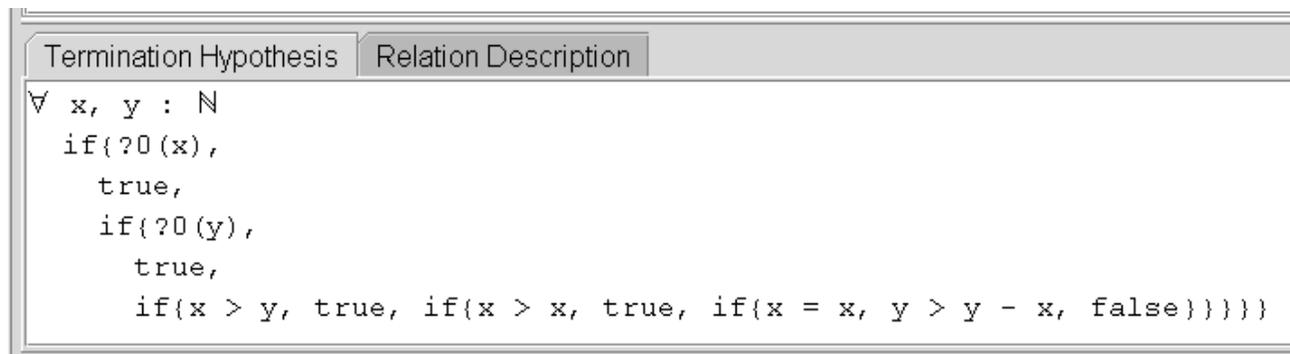
- Mit Maus Terminierungshypothese selektieren und dann *Doppelclick* oder *Proof* “drücken” um deren Beweisbaum im *Proof Window* anzuzeigen.
 - *Grünes Icon* = Status *verified* = Terminierungshypothese ist *bewiesen*.
 - *Blaues Icon* = Status *ready* = Terminierungshypothese *nicht bewiesen* (\Rightarrow Beweisbaum mittels *HPL*-Regeln editieren).
 - *Rotes Icon* = Status *disproved* = Terminierungshypothese ist *widerlegt* (\Rightarrow anderen Maßterm verwenden)



- Der *rekursive Aufruf*, für den die *selektierte Terminierungshypothese* erzeugt wurde, wird *gelb unterlegt* angezeigt

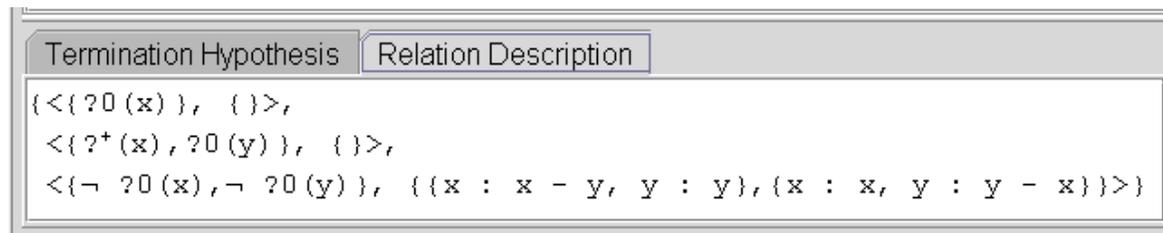
```
function gcd(x : ℕ, y : ℕ) : ℕ <=
if ?0(x)
  then y
  else if ?0(y)
    then x
    else if x > y
      then gcd(x - y, y)
      else gcd(x, y - x)
    end_if
  end_if
end_if
```

- Der *PrettyPrint* der selektierten Terminierungshypothese wird unter dem Reiter *Termination Hypothesis* angezeigt



```
Termination Hypothesis  Relation Description
∀ x, y : ℕ
  if(?0(x),
    true,
    if(?0(y),
      true,
      if(x > y, true, if(x > x, true, if(x = x, y > y - x, false))))))
```

- Konnten *alle Terminierungshypothesen* (zu einem Maßterm oder einer Liste von Maßtermen) *bewiesen* werden, so wird für die Prozedur eine *optimierte Relationenbeschreibung* erzeugt und unter dem Reiter *Relation Description* angezeigt



```
Termination Hypothesis  Relation Description
<{?0(x)}, {}>,
<{?+(x), ?0(y)}, {}>,
<{¬ ?0(x), ¬ ?0(y)}, {{x : x - y, y : y}, {x : x, y : y - x}}>
```

- Mit
 - *Close* wird das *Termination Window* geschlossen
 - *Recompute* wird eine *automatische Terminierungsanalyse* gestartet
 - *Show Termination Analysis* werden *Details* der *automatischen Terminierungsanalyse* angezeigt



Bemerkung 6

- **System-Bug:** Bei Selektion einer Terminierungshypothese im *Termination Window* kann es zu *Fehlermeldungen* im *System Log* kommen – in diesem Fall einfach *Termination Window* über *Close* schließen und anschließend erneut öffnen.
- **Behauptung:** “Falls alle Terminierungshypothesen, die zu einem Maßterm erzeugt wurden, widerlegt sind, so terminiert die Prozedur nicht.”
Überlegen: Stimmt das (\Rightarrow Beweis) oder stimmt das nicht (\Rightarrow Gegenbeispiel).