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] $>_{R_{\mathtt{proc}'},x_1\ldots x_k}$ 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 *fundierten* Relation garantiert *Terminierung*.
- Aus einer *terminierenden* Prozedur gewinnt man (uniform) eine *fundierte* Relation.

Satz 2 (Fundierte Relationenbeschreibungen und terminierenden Prozeduren)
Die Relationenbeschreibung $R_{\mathtt{p}}$ *einer* $\mathcal{L}$*-Prozedur* $\mathtt{p}$ *ist* fundiert *gdw. die* $\mathcal{L}$*-Prozedur* $\mathtt{p}$ terminiert.

Beweis: *$R_{\mathtt{p}}$ ist fundiert gdw. jede monomorphe Instanz $R_{\mathtt{p}'}$ von $R_{\mathtt{p}}$ fundiert ist (=> **Kapitel 6**, Definition 8) gdw. jede Relation $>_{R_{\mathtt{p}'},x_1\ldots x_k}$ fundiert ist (=> **Kapitel 6**, Definition 8) gdw. gdw. $\mathtt{p}$ terminiert (=> 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 $\mathtt{p}$ erzeugt das System eine Menge von *Terminierungshypothesen* $th_1, \ldots, th_n$ für $\mathtt{p}$.
- Terminierungshypothesen haben die gleiche *Form wie Lemmata*.
- Das System versucht alle *Terminierungshypothesen* zu *verifizieren*.
- **Es gilt:** Sind alle *Terminierungshypothesen* $th_1, \ldots, th_n$ *wahr*, so *terminiert* die *Prozedur* $\mathtt{p}$.

- **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*

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

  wobei AND($\{b_1, \ldots, b_n\}$) $=$ 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 $q_0, q_1, q_2, \ldots$ von Konstruktorgrundtermen vom Typ `nat`, so daß für alle $i \in \mathbb{N}$ gilt:
  Bei Berechnung von $\mathtt{p(}q_i\mathtt{)}$ wird $\mathtt{p(}q_{i+1}\mathtt{)}$ rekursiv aufgerufen.
- Damit gilt $eval_P(\text{AND}(H\,[\mathtt{x}/q_i])) = \mathtt{true}$ für alle $i \in \mathbb{N}$, denn die Bedingung AND($H\,[\mathtt{x}/q_i]$), die zum jeweils nächsten rekursiven Aufruf führt, muß ja gelten.
- Damit gilt $eval_P(q_i > t\,[\mathtt{x}/q_i]) = \mathtt{true}$ für alle $i \in \mathbb{N}$, denn th_A ist nach Voraussetzung verifiziert.
- Folglich gilt $q_i > eval_P(t\,[\mathtt{x}/q_i])$ für alle $i \in \mathbb{N}$.
- Für $r_0 := q_0$ und $r_{i+1} := eval_P(t\,[\mathtt{x}/r_i])$ für alle $i \in \mathbb{N}$ ist $r_0, r_1, r_2, \ldots$ 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. $\blacksquare$

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

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

und damit die (trivial zu verifizierende) Terminierungshypothese[4,5]

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

---

[4] Nach Definition müßten wir eigentlich $\forall$ x:nat if{if{$\neg$?0(x), $\neg$?0($^-$(x)), false}, x $>$ $^-$($^-$(x)), true} schreiben. Wir verwenden jedoch in den Beispielen die *normalisierte* Form (1) (d.h. $a \to (b \to c)$ anstatt $a \wedge b \to c$), 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

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

    die Terminierungshypothesen

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

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

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

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

und damit die (trivial zu verifizierenden) Terminierungshypothesen[6]

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

sowie

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

[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 $\tau \neq$ `nat`
- *Dann*:
  – Definiere eine *Terminierungsfunktion* $m : \tau \to$ `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

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

  – Verifiziere die Terminierungshypothese th_A.

Beispiel 3

Für die Prozedur

```
function minsort(k:list[nat]):list[nat] <=
if ?ø(k)
 then ø
 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 ?ø(l) then 0 else +(| tl(l) |) end_if
```

Für `minsort` *erhält man die rekursive atomare Relationenbeschreibung*

$$A = \langle \{\neg ?\varnothing(k)\}, \{\{k/k \backslash minimum(k)\}\} \rangle$$

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

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

Bemerkung 1

(1) In Beispiel 3 wird die *Terminierung* von $|\cdots|$ *vorausgesetzt*; dies ist erlaubt, denn der *Terminierungsnachweis* von $|\cdots|$ 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 (=> Übung).

(3) Genaugenommen haben wir schon in den Beispielen 1 und 2 Maßterme verwendet (=> *Überlegen*: Wie lauten die Maßterme dort?).

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

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

    die Terminierungshypothese $th_A =$

    $$\forall\, \mathtt{x}_1\mathtt{:}\tau_1, \ldots, \mathtt{x}_n\mathtt{:}\tau_n\ \mathtt{if}\{\text{AND}(H)\mathtt{,}\ \mathtt{m} > \mathtt{m}\,[\mathtt{x}_1/\mathtt{t}_1, \ldots, \mathtt{x}_n/\mathtt{t}_n]\,\mathtt{,}\ \mathtt{true}\}.$$
  - 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 $\texttt{x} + \texttt{y}$ *und erhalten die Terminierungshypothesen*

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

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

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

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

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* (=> 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. (=> *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

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

sowie

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

Als Maßterm wählen wir $|\,\mathtt{a}\,|$ `- i` *(und ignorieren damit den formalen Parameter* `x`*). Damit erhält man die Terminierungshypothesen*

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

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

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

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

Bemerkung 3

(1) In Beispiel 5 wird die *Terminierung* von $|\cdots|$ *vorausgesetzt*; dies ist erlaubt, denn der *Terminierungsnachweis* von $|\cdots|$ 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 $\texttt{p}$ besitzt $n \geq 1$ formale Parameter $\texttt{x}_1, \ldots, \texttt{x}_n$ der Typen $\tau_1, \ldots, \tau_n$
- *Dann*:
  - Definiere *Terminierungsfunktionen* $m_1, \ldots, m_j : \tau_1 \times \ldots \times \tau_n \to \texttt{nat}$ durch Angabe einer *Liste von Maßtermen* $\texttt{m}_1, \ldots, \texttt{m}_j$ jeweils vom Typ $\texttt{nat}$, die nur $\texttt{x}_1, \ldots, \texttt{x}_n$ als Variable enthalten.
  - Bilde für *jede* atomare rekursive Relationenbeschreibung

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

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

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

  - 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*

$th_{A_1} =$

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

sowie $th_{A_2} =$

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

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)))),
         other : cons(car(x), flatten(cdr(x)))
        end_case,
 other : x
end_case
```

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

$$A_1 = \begin{array}{l} \langle \{?cons(x), ?cons(car(x))\}\,, \\ \quad \{\{x/cons(car(car(x)), cons(cdr(car(x)), cdr(x)))\}\}\rangle \end{array}$$

sowie

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

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

$$
\begin{array}{l}
th_{A_1} = \\
\forall\, \texttt{x:sexpr if}\{\texttt{?cons(x)}, \\
\qquad\qquad \texttt{if}\{\texttt{?cons}(\texttt{car}(\texttt{x})), \\
\qquad\qquad\quad \texttt{if}\{\texttt{\#nodes(x)} > \texttt{\#nodes(cons(}\ldots\texttt{))}, \\
\qquad\qquad\qquad \texttt{true}, \\
\qquad\qquad\qquad \texttt{if}\{\texttt{\#nodes(x)} = \texttt{\#nodes(cons(}\ldots\texttt{))}, \\
\qquad\qquad\qquad\quad \texttt{\#nodes(car(x))} > \texttt{\#nodes(car(cons(}\ldots\texttt{)))}, \\
\qquad\qquad\qquad\quad \texttt{false}\}\}, \\
\qquad\qquad\quad \texttt{true}\}, \\
\qquad\qquad \texttt{true}\}
\end{array}
$$

sowie

$th_{A_2} =$

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

Bemerkung 4

(1) In Beispiel 7 wird die *Terminierung* der Prozedur `#nodes` (=> **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 $\Re_1, \ldots, \Re_h$ 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 (=> **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 ?ø(k)
 then k
 else if ?0(n)
       then k
       else ntl(⁻(n), tl(k))
      end_if
end_if
```

erhält man die rekursive atomare Relationenbeschreibung

$$A = \left\langle \{\neg ?\text{ø}(\mathtt{k}), \neg ?0(n)\}, \left\{\left\{n/^{-}(n), k/tl(k)\right\}\right\}\right\rangle .$$

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

$$\mathtt{if}\{\neg ?\text{ø}(\mathtt{k}), \mathtt{if}\{\neg ?0(\mathtt{n}), \mathtt{n} > {}^{-}(\mathtt{n}), \mathtt{true}\}, \mathtt{true}\}$$

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

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

- Bei Beweis von $th_{A,1}$ wird die Bedingung "$\neg\texttt{?ø(k)}$" nicht verwendet. Damit erhält man die *optimierte Relationenbeschreibung* (=> **Kapitel 7**, Abschnitt 3)

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

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

$$R^{(2)}_{\texttt{ntl}} := \left\{ \begin{array}{l} \langle\{\texttt{?ø(}k\texttt{)}\}\,, \emptyset\rangle\,, \\ \langle\{\neg\texttt{?ø(}k\texttt{)}\}\,, \{\{k/\texttt{tl(}k\texttt{)}\}\}\rangle \end{array} \right\}\,.$$

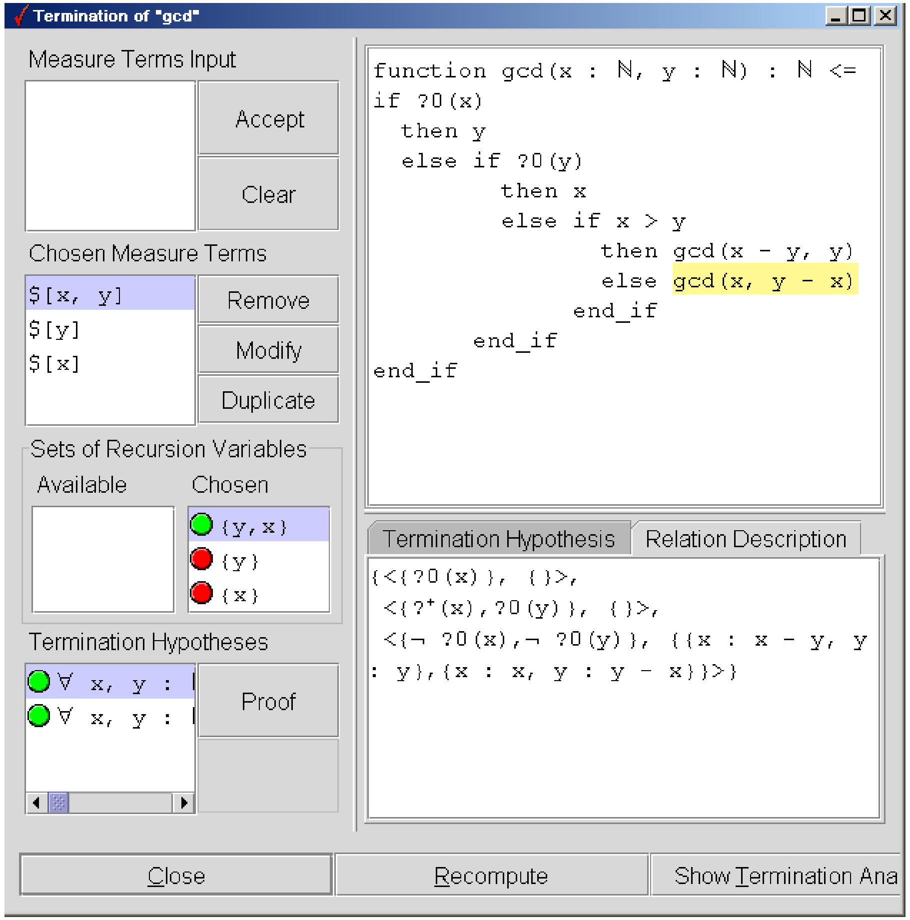
Bemerkung 5

(1) Anstatt Maßterme können auch Listen von Maßtermen der Reihe nach angegeben werden.

(2) Beide *Terminierungsnachweise* von `ntl` gelingen *automatisch* (=> ausprobieren !); interaktiv hier nur zur Illustration.

4 Benutzerinteraktion bei Terminierungsbeweisen

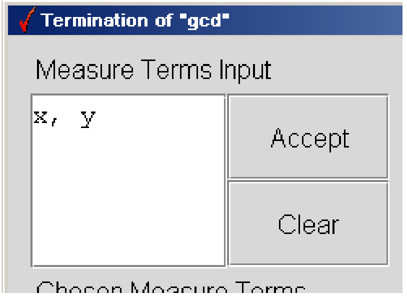
- Interaktive Terminierungsbeweise werden in ***VeriFun*** über das *Termination Window* analysiert und gesteuert
- **Aufruf:**
  1. Selektiere Prozedur-Icon im *Programm Window*
  2. Öffnen des *Termination Window* über
     * Maus Doppelklick = Menue *Program**Set Termination* (falls Prozedur noch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



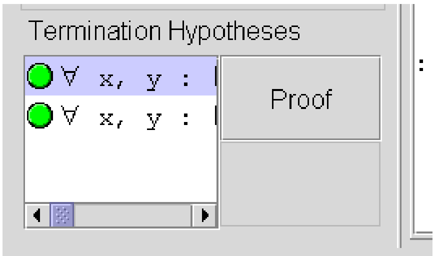
- Versuch automatischer Terminierungsbeweis mit Maßterm x
  – gescheitert, da (eine) Terminierungshypothese widerlegt werden kann
- Versuch automatischer Terminierungsbeweis mit Maßterm y
  – gescheitert, da (eine) Terminierungshypothese widerlegt werden kann
- Versuch automatischer Terminierungsbeweis 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 *Doppelklick* oder *Proof* “drücken” um deren Beweisbaum im *Proof Window* anzuzeigen.
  - *Grünes* Icon = Status *verified* = Terminierungshypothese ist *bewiesen*.
  - *Blaues* Icon = Status *ready* = Terminierungshypothese *nicht bewiesen* (=> Beweisbaum mittels *HPL*-Regeln editieren).
  - *Rotes* Icon = Status *disproved* = Terminierungshypothese ist *widerlegt* (=> anderen Maßterm verwenden)



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

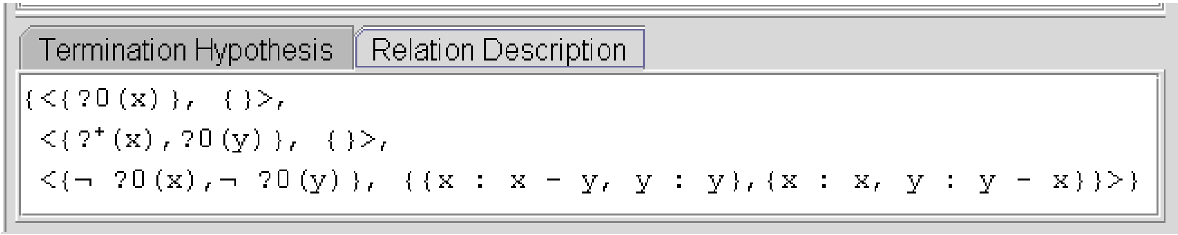
```
function gcd(x : ℕ, y : ℕ) : ℕ <=
if ?0(x)
  then y
  else if ?0(y)
         then x
         else if x > y
                then gcd(x - y, y)
                else gcd(x, y - x)
              end_if
       end_if
end_if
```

- 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



- 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 (=> Beweis) oder stimmt das nicht (=> Gegenbeispiel).