

Automatisierte Logik und Programmierung

Prof. Chr. Kreitz

Universität Potsdam, Theoretische Informatik — Sommersemester 2004

Blatt 2 — Abgabetermin: —

Aufgabe 2.1

Beweisen Sie mit und ohne Verwendung der `arith`-Entscheidungsprozedur

$$\forall x, y: \mathbb{Z}. \quad x < y + 4 \wedge y - 23 < 55 \wedge x > y \wedge y - 1 > 35 \Rightarrow (x < 86 \vee y < 1) \wedge 25 < x$$

Hinweis: Verwenden Sie die Strategie `simple_prover` um das Beweisziel vorzubereiten. Gehen Sie davon aus, daß `Arith` – die Taktik-Version von `arith` – auch die in Definition Ungleichungen $\geq$, $\leq$, $\neq$ verarbeiten kann und `Wf` alle anfallenden Wohlgeformtheitsziele lösen kann.

Für die Lösung ohne `Arith` empfiehlt es sich, allgemeine arithmetische Lemmata aufzustellen und mit `InstLemma` (Manual, Seite 128) anzuwenden. `InstLemma`: `tok -> term list -> tactic` instantiiert Lemmata der Form $\forall x_1:T_1 \dots x_n:T_n. P \Rightarrow Q$ mit den gelisteten Termen und wendet dann `modus ponens (impE)` an. Die Lemmata brauchen nicht bewiesen zu werden.

Aufgabe 2.2 (Grundlagen der `arith`-Prozedur)

- 2.2-a Beschreiben Sie einen Algorithmus, der überprüft, ob ein gerichteter Graph einen positiven Zyklus besitzt.
- 2.2-b Zeigen Sie, daß Entscheidbarkeit von Formeln unter aussagenlogischen Konnektiven ($\neg$, $\wedge$, $\vee$, $\Rightarrow$) abgeschlossen ist.
- 2.2-c Beschreiben Sie (informal) eine Taktik, welche elementar-arithmetische Formeln in konjunktive Normalform (als Vorbereitung für `arith`) umwandelt.

Aufgabe 2.3 (Etwas Theoretisches als Zusatzaufgabe)

- 2.3-a Zeigen Sie, daß eine Sequenz $\Gamma \vdash G_1 \vee \dots \vee G_n$ genau dann gültig ist, wenn die Sequenz $\Gamma, \neg G_1, \dots, \neg G_n \vdash \text{ff}$ gültig ist, sofern alle Formeln G_i entscheidbar sind.
- 2.3-b Es sei $\Gamma = v_1 \geq u_1 + c_1, \dots, v_n \geq u_n + c_n$ eine Menge von atomaren arithmetischen Formeln, wobei die v_i und u_i Variablen (oder 0) und die c_i ganzzahlige Konstanten sind, und $\mathcal{G}$ der Graph, der die Ordnungsrelation zwischen den Variablen von Γ beschreibt.
Zeigen Sie, daß Γ genau dann widersprüchlich ist, wenn $\mathcal{G}$ einen positiven Zyklus besitzt.

Aufgabe 2.4

Beweisen Sie die Gültigkeit der Formel

$$\forall a, b: \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z}. \forall c, d, e, f: \mathbb{Z}. \quad d = e \Rightarrow a(b(d, f), c) = a(b(e, f), c)$$

ohne Verwendung der `equality` Regel.

Hinweis: Verwenden Sie Tacticals wie `THEN`, `Repeat` etc. und gehen Sie davon aus, daß die Tactic `Wf` alle hierbei anfallenden Wohlgeformtheitsziele lösen kann.

Lösung 2.0 Mit dem sechsten Übungsblatt sollen die Vorteile und Schwierigkeiten von Entscheidungsprozeduren kennengelernt werden.

Lösung 2.1 Ziel dieser Aufgabe ist es, den praktischen Nutzen zu erkennen, den Entscheidungsprozeduren schon in relativ leichten Problemstellungen bringen

Ohne Verwendung der `arith`-Regel wird die Lösung sehr aufwendig. Wir müssen die in `arith` enthaltene Gesetze als Lemmata formulieren (und beweisen), um sie instantiieren zu können.

```

⊢ ∀x,y:ℤ. x<y+4 ∧ y-23<55 ∧ x>y ∧ y-1>35 ⇒ (x<86 ∨ y<1) ∧ 25<x
BY simple_prover
|
| \
|  x:ℤ, y:ℤ, x<y+4, y-23<55, x>y, y-1>35 ⊢ x<86 ∨ y<1
|  BY orI1
|  |
|  | \
|  |  ... ⊢ x<86
|  |  BY InstLemma 'sub_1' [y,23,55]
|  |  |
|  |  | \
|  |  |  ..., y<55+23 ⊢ x<86
|  |  |  BY InstLemma 'mon_add' [y,55+23,4]
|  |  |  |
|  |  |  | \
|  |  |  |  ..., y<55+23, y+4<55+23+4 ⊢ x<86
|  |  |  |  BY InstLemma 'trans_1' [x,y+4,55+23+4]
|  |  |  |  |
|  |  |  |  | \
|  |  |  |  |  ..., y<55+23, y+4<55+23+4, x<55+23+4 ⊢ x<86
|  |  |  |  |  BY InstLemma 'const_1' []
|  |  |  |  |  |
|  |  |  |  |  | \
|  |  |  |  |  |  ..., y<55+23, y+4<55+23+4, x<55+23+4, 55+23+4<86 ⊢ x<86
|  |  |  |  |  |  BY InstLemma 'trans_1' [x,55+23+4,86]
|  |  |  |  |  |  |
|  |  |  |  |  |  | \
|  |  |  |  |  |  |  ..., y<55+23, y+4<55+23+4, x<55+23+4, 55+23+4<86, x<86 ⊢ x<86
|  |  |  |  |  |  |  BY hypothesis 11
|  |  |  |  |  |  |
|  |  |  |  |  |  | \
|  |  |  |  |  |  |  x:ℤ, y:ℤ, x<y+4, y-23<55, x>y, y-1>35 ⊢ 25<x
|  |  |  |  |  |  |  BY InstLemma 'sub_2' [y,1,35]
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | \
|  |  |  |  |  |  |  |  ..., y>35+1 ⊢ 25<x
|  |  |  |  |  |  |  |  BY InstLemma 'trans_2' [x,y,35+1]
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  | \
|  |  |  |  |  |  |  |  |  ..., y>35+1, x>35+1 ⊢ 25<x
|  |  |  |  |  |  |  |  |  BY InstLemma 'const_2' []
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  | \
|  |  |  |  |  |  |  |  |  |  ..., y>35+1, x>35+1, 35+1>25 ⊢ 25<x
|  |  |  |  |  |  |  |  |  |  BY InstLemma 'trans_2' [x,35+1,25]
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  | \
|  |  |  |  |  |  |  |  |  |  |  ..., y>35+1, x>35+1, 35+1>25, x>25 ⊢ 25<x
|  |  |  |  |  |  |  |  |  |  |  BY InstLemma 'less' [x,25]
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  | \
|  |  |  |  |  |  |  |  |  |  |  |  ..., y>35+1, x>35+1, 35+1>25, x>25, 25<x ⊢ 25<x
|  |  |  |  |  |  |  |  |  |  |  |  BY hypothesis 10

```

Die folgenden Lemmata spielen hierbei eine Rolle

trans_1:	$\forall x,y,z:\mathbb{Z}. x<y \wedge y<z \Rightarrow x<z$	mon_add:	$\forall x,y,z:\mathbb{Z}. x<y \Rightarrow x+z<y+z$
trans_2:	$\forall x,y,z:\mathbb{Z}. x>y \wedge y>z \Rightarrow x>z$	less:	$\forall x,y:\mathbb{Z}. x>y \Rightarrow y<x$
sub_1:	$\forall x,y,z:\mathbb{Z}. x-y<z \Rightarrow x<z+y$	const_1:	$55+23+4 < 86$
sub_2:	$\forall x,y,z:\mathbb{Z}. x-y>z \Rightarrow x>z+y$	const_2:	$35+1 > 25$

Mit Arith dagegen ist die Lösung natürlich sehr kurz.

```

⊢ ∀x,y:ℤ.  x<y+4 ∧ y-23<55 ∧ x>y ∧ y-1>35 ⇒ (x<86 ∨ y<1) ∧ 25<x
BY simple_prover
| \
| x:ℤ, y:ℤ, x<y+4, y-23<55, x>y, y-1>35 ⊢ x<86 ∨ y<1
| BY Arith THEN Wf
| \
| x:ℤ, y:ℤ, x<y+4, y-23<55, x>y, y-1>35 ⊢ 25<x
BY Arith THEN Wf

```

Lösung 2.2 Ziel dieser Aufgabe ist es, bekannte Ergebnisse aus verschiedenen Teilgebieten der Informatik – Grundalgorithmen, theoretische Informatik, Logik & Schaltalgebra – im Sinne eines neuen Verwendungszwecks zusammenzusetzen.

2.2–a Die Grundidee des Algorithmus ist, statt nach Zyklen zu suchen und diese zu überprüfen einfach die maximalen Gewichte von Pfaden zu bestimmen, die in einem Knoten v_i beginnen und in einem Knoten v_j des Graphen enden. Anschließend prüft man, ob das maximale Gewicht eines Pfades von v_i nach v_i für ein i positiv ist.

Diese Berechnung stützt sich auf einen bekannten Standardalgorithmus. Man definiere $C_{i,j}^k$ als das maximale Gewicht eines Pfades von v_i nach v_j bei dem (außer v_i und v_j selbst) nur die Knoten $v_1..v_k$ durchlaufen werden dürfen.

Dabei ist $C_{i,j}^0$ das Gewicht der Kante von v_i nach v_j , falls eine solche existiert, und $-\infty$ andernfalls. Man macht sich schnell klar, daß $C_{i,j}^{k+1}$ entweder identisch ist mit $C_{i,j}^k$ oder das maximale Gewicht eines Pfades von v_i nach v_j , der über v_k läuft – also die Summe von $C_{i,k}^k$ und $C_{k,j}^k$, je nachdem, welches Gewicht höher ist.

Integriert man nun den Test, ob ein positiver Zyklus gefunden wurde, in den Algorithmus, so ergibt sich folgendes Programmstück (in ALGOL-ähnlicher Notation)

```

FOR i:=1 TO n DO
  FOR j:=1 TO n DO
     $A_{i,j} := C_{i,j}^0$ 
  END
END;
poscycle := false;

FOR k:=1 TO n WHILE NOT poscycle DO
  FOR i:=1 TO n WHILE NOT poscycle DO
    FOR j:=1 TO n WHILE NOT poscycle DO
       $A_{i,j} := \max(A_{i,j}, A_{i,k} + A_{k,j});$ 
      poscycle := j=i AND  $A_{i,j} > 0$ 
    END
  END
END

```

Weitere Details findet man im Artikel “An algorithm for checking PL/CV arithmetic inferences” (siehe [Constable *et.al.*1982, Appendix D]) auf Seite 241ff.

2.2–b Der Beweis ist ein Standardresultat der theoretischen Informatik: Wir müssen nur zeigen, daß aus der Entscheidbarkeit von beliebigen Formeln A und B die Entscheidbarkeit von $A \wedge B$, $A \vee B$, $A \Rightarrow B$ und $\neg A$ folgt.

Eine Formel A mit freien Variablen $x_1, \dots, x_n$ ist entscheidbar, wenn es eine berechenbare (totale!) Funktion $\chi_A: \mathbb{N}^n \rightarrow \mathbb{N}$ gibt mit der Eigenschaft

$$\chi_A(x_1, \dots, x_n) = 0 \text{ genau dann, wenn } A[x_1, \dots, x_n] \text{ gültig ist.}$$

χ_A ist dann die *charakteristische Funktion* von A . (Man könnte auch $\chi_A(x_1, \dots, x_n) = 1$ fixieren, aber mit 0 ist es praktischer).

Es seien χ_A und χ_B die charakteristischen Funktionen von A und B mit (o.B.d.A.) freien Variablen $x_1, \dots, x_n$. Dann ergeben sich folgende charakteristische Funktionen

$$A \wedge B: \lambda x_1, \dots, x_n. \chi_A(x_1, \dots, x_n) + \chi_B(x_1, \dots, x_n)$$

$$A \vee B: \lambda x_1, \dots, x_n. \chi_A(x_1, \dots, x_n) * \chi_B(x_1, \dots, x_n)$$

$$A \Rightarrow B: \lambda x_1, \dots, x_n. (1 - \chi_A(x_1, \dots, x_n)) * \chi_B(x_1, \dots, x_n)$$

$$\neg A: \lambda x_1, \dots, x_n. 1 - \chi_A(x_1, \dots, x_n) \quad (\text{Man beachte, daß } x-y=0 \text{ ist, wenn } x < y \text{ gilt.})$$

Dies ist der *Beweis* dafür, daß die Theorie $\mathcal{A}$ prinzipiell entscheidbar ist. Prinzipiell ist hiermit auch ein Verfahren zur Überprüfung elementar-arithmetischer Formeln gegeben. Dieses aber wäre extrem ineffizient. Beweis und Entscheidungsverfahren sind verschiedene Aspekte: ein einfacher Beweis liefert oft nur ein komplexes Verfahren und umgekehrt.

2.2-c Im Prinzip müsste die Taktik einfach nur die folgenden Konversionen der Reihe nach rekursiv anwenden.

$A \Rightarrow B \quad \mapsto \neg A \vee B$	$\neg(s=t) \mapsto s \neq t$
$\neg(A \wedge B) \mapsto \neg A \vee \neg B$	$\neg(s \neq t) \mapsto s=t$
$\neg(A \vee B) \mapsto \neg A \wedge \neg B$	$\neg(s < t) \mapsto s \geq t$
$A \vee (B \wedge C) \mapsto A \vee B \wedge A \vee C$	$\neg(s > t) \mapsto s \leq t$
$\neg \neg A \mapsto A$	$\neg(s \geq t) \mapsto s < t$
	$\neg(s \leq t) \mapsto s > t$

Im Detail ist dies aber relativ mühsam, da eine Substitutionsregel für logische Äquivalenzen nicht existiert. So muß man sich damit behelfen, die Substitution separat auszurechnen und die modifizierte Formel mit der Regel *cut* einzuführen.

Man betrachtet hierzu die Formel von außen und analysiert ihre Struktur. Im ersten Schritt sucht man in der Konklusion C nach Teilformeln der Gestalt $A \Rightarrow B$, sobald eine solche gefunden wird, ersetzt man diese durch $\neg A \vee B$. Man erhält somit eine neue Formel F und ruft die Regel *cut* (-1) F auf.

Im ersten Teilziel ist nun $\vdash F$ zu zeigen und man wiederholt den die Konversion bis keine Formeln der Teilformeln der Gestalt $A \Rightarrow B$ mehr da sind. Anschließend konvertiert man Teilformeln der Gestalt $\neg(A \wedge B)$ usw.

Im zweiten Teilziel muß man nun $F \vdash C$ zeigen. Hierzu kann man die Taktik **simple_prover** einsetzen, die genau zu dem Teilziel $\neg A \vee B \vdash A \Rightarrow B$ führen wird, da F und C bis auf diese Teilformeln ja identisch sind und nur zerlegt werden müssen.

Nun benötigt man ein Lemma der Gestalt

$$\forall A, B: 1. \quad A \text{ entscheidbar} \wedge B \text{ entscheidbar} \quad \Rightarrow \quad A \Rightarrow B \Leftrightarrow \neg A \vee B$$

sowie Lemmata über die Entscheidbarkeit elementar-arithmetischer Formeln gemäß Teil b). Damit läßt sich dann das restliche Teilziel beweisen

Dieses Verfahren ist natürlich völlig ineffizient und wird in der Praxis niemals Einzug finden. Jedoch ist hiermit gezeigt, daß `arith` – die ja nur auf Disjunktionen atomarer Formeln anwendbar ist – zu einer vollständigen Entscheidungsverfahren für die gesamte Theorie $\mathcal{A}$ ergänzt werden kann.

In der Praxis ist es viel effizienter, auf die Normalisierung zu verzichten und die zu untersuchenden Formeln von Hand freizulegen. Im Prinzip braucht man dazu nur die Tactic `prover` (siehe Übung 5.5) so zu erweitern, daß in `simple_prover` der Aufruf von `arith` früh integriert wird. Hierdurch werden Implikationen, Negationen und Konjunktionen automatisch zerlegt und Disjunktionen separat verfolgt.

Lösung 2.3 Ziel dieser Aufgabe ist es, die theoretischen Fundamente der `arith`-Prozedur zu überprüfen.

- 2.3–a Ein Standard-Ergebnis der Logik, bei dem die Entscheidbarkeit sich in der Gültigkeit von $G_1 \vee \dots \vee G_n \Leftrightarrow \neg \neg (G_1 \vee \dots \vee G_n) \Leftrightarrow \neg (\neg G_1 \wedge \dots, \neg G_n)$ niederschlägt. Der Rest entspricht dann der Einführungsregel für $\neg$.
- 2.3–b Der Beweis ist verhältnismäßig aufwendig, wenn er im Detail geführt werden muß. Details findet man im Artikel “*An algorithm for checking PL/CV arithmetic inferences*” (siehe [Constable *et.al.*1982, Appendix D]) auf Seite 238ff.

Literatur

[Constable *et.al.*1982] Robert L. Constable, Scott D. Johnson, and Carl D. Eichenlaub. *Introduction to the PL/CV2 Programming Logic*, volume 135 of *Lecture Notes in Computer Science*. Springer Verlag, Berlin, Heidelberg, New-York, 1982.

Lösung 2.4 Ziel dieser Aufgabe ist es, einen Gleichheitsbeweis mit den normalen Regeln zu führen und hierdurch die Vorteile von `equality` zu erkennen

$\vdash \forall a, b: \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z}. \forall c, d, e, f: \mathbb{Z}. \quad d = e \in \mathbb{Z} \Rightarrow a(b(d, f), c) = a(b(e, f), c) \in \mathbb{Z}$

BY Repeat allI THEN impI

| $a: \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z}, b: \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z}, c: \mathbb{Z}, d: \mathbb{Z}, e: \mathbb{Z}, f: \mathbb{Z}, d = e \in \mathbb{Z}$

| $\vdash a(b(d, f), c) = a(b(e, f), c) \in \mathbb{Z}$

BY subst 1 $d = e \in \mathbb{Z} \quad x \quad a(b(x, f), c) = a(b(e, f), c) \in \mathbb{Z}$

| $\dots, d = e \in \mathbb{Z} \quad \vdash \quad d = e \in \mathbb{Z}$

| BY hypothesis 7

| $\dots, d = e \in \mathbb{Z} \quad \vdash \quad a(b(e, f), c) = a(b(e, f), c) \in \mathbb{Z}$

| BY applyEq $\mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z}$

die Taktik `Wf` löst all dies ebenfalls

| $\dots, \vdash a = a \in \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z}$

| BY hypEq 1

| $\dots, \vdash (b(e, f), c) = (b(e, f), c) \in \mathbb{Z} \times \mathbb{Z}$

| BY pairEq_indep

| $\dots \quad \vdash \quad b(e, f) = b(e, f) \in \mathbb{Z}$

```

|   | BY applyEq  $\mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z}$ 
|   | | \
|   | |   ...  $\vdash b=b \in \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z}$ 
|   | | BY hypEq 2
|   | | \
|   | |   ...  $\vdash (e,f)=(e,f) \in \mathbb{Z} \times \mathbb{Z}$ 
|   | | BY pairEq_indep
|   | | | \
|   | | |   ...  $\vdash e=e \in \mathbb{Z}$ 
|   | | | BY hypEq 5
|   | | | \
|   | | |   ...  $\vdash f=f \in \mathbb{Z}$ 
|   | | | BY hypEq 5
|   | | \
|   | |   ...  $\vdash c=c \in \mathbb{Z}$ 
|   | | BY hypEq 3
|   | \
|   |   ...  $\vdash d=e \in \mathbb{Z}, x:\mathbb{Z} \vdash (a(b(x,f),c)=a(b(e,f),c) \in \mathbb{Z}) \in 1$ 
|   BY Wf

```

Die Vorgehensweise ist ganz schematisch und läßt sich leicht programmieren, da im wesentlichen Terme innerhalb von Gleichheiten zerlegt werden (Dekomposition). Der Name der Regel ergibt sich i.w. aus dem Operatornamen des Terms. So ist z.B. $a(b(e,f),c)$ ein Term mit Namen `apply`, also wird `applyEq` angewandt. Der Parameter dieser Regel ist der Typ des ersten Teilterms, also der von a , der normalerweise in den Hypothesen zu finden ist.

Wenn man sich diese – intuitiv naheliegende – Lösung genauer ansieht, stellt man fest, daß die Substitution zu Beginn eigentlich überflüssig ist, da man die Terme ohnehin zerlegen muß. Dies führt zu folgender optimierter Lösung:

```

 $\vdash \forall a,b:\mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z}. \forall c,d,e,f:\mathbb{Z}. d=e \in \mathbb{Z} \Rightarrow a(b(d,f),c)=a(b(e,f),c) \in \mathbb{Z}$ 
BY Repeat allI THEN impI
| \
|    $a:\mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z}, b:\mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z}, c:\mathbb{Z}, d:\mathbb{Z}, e:\mathbb{Z}, f:\mathbb{Z}, d=e \in \mathbb{Z}$ 
|    $\vdash a(b(d,f),c)=a(b(e,f),c) \in \mathbb{Z}$ 
|   BY applyEq  $\mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z}$ 
|   | \
|   |   ...  $\vdash a=a \in \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z}$ 
|   | BY hypEq 1
|   | \
|   |   ...  $\vdash (b(d,f),c)=(b(e,f),c) \in \mathbb{Z} \times \mathbb{Z}$ 
|   | BY pairEq_indep
|   | | \
|   | |   ...  $\vdash b(d,f)=b(e,f) \in \mathbb{Z}$ 
|   | | BY applyEq  $\mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z}$ 
|   | | | \
|   | | |   ...  $\vdash b=b \in \mathbb{Z} \times \mathbb{Z} \rightarrow \mathbb{Z}$ 
|   | | | BY hypEq 2
|   | | | \
|   | | |   ...  $\vdash (d,f)=(e,f) \in \mathbb{Z} \times \mathbb{Z}$ 
|   | | | BY pairEq_indep
|   | | | | \
|   | | | |   ...  $\vdash d=e \in \mathbb{Z}$ 
|   | | | | BY hypothesis 7
|   | | | | \
|   | | | |   ...  $\vdash f=f \in \mathbb{Z}$ 
|   | | | | BY hypEq 5
|   | | | \
|   | | |   ...  $\vdash c=c \in \mathbb{Z}$ 
|   | | | BY hypEq 3

```

die Taktik Wf löst all dies von selbst