Nichtlineare Gleichungssysteme I

Die Signatur:
Menge $\Re$ der reellen Zahlen,
arithmetische Funktionssymbole
$+,\ *,\ \log,\ \sin,\ \exp, \ldots$ sowie ':',
die Constraintsymbole $=_I, <, \leq, >, \geq, \neq_I$ und
'::'.

Die erlaubten Constraints:
$C ::= \mathit{true} \mid \mathit{false} \mid X :: n : m \mid s \odot t \mid C \wedge C$
wobei n und m reelle Zahlen sind, s und t
beliebige arithmetische Ausdrücke über der
Signatur von I sind und $\odot$ eines der
Constraintsymbole $=_I, <, \leq, >, \geq, \neq_I$ ist.

Die Constrainttheorie:
Beschreibt die arithmetischen Funktionen und
Relationen (Erweiterung von Tarskis Theorie),
inklusive:

$$X :: n : m \leftrightarrow n \leq X \wedge X \leq m$$

Anmerkung

Auf die rationalen Zahlen kann man verzichten, da die Ungenauigkeit der Maschinendarstellung reeller Zahlen als Fließkommazahlen durch Intervalle beschrieben werden kann, und da nichtlineare und trigonometrische Funktionen auch irrationale Zahlen erzeugen können. $\sqrt{2}$ läßt sich zum Beispiel durch das Intervallconstraint `Sqrt2::1.41:1.42` eingrenzen und π durch `Pi::3.14:3.15`.

Stand 1998

Zwei Ansätze

Erweiterung der Variablenelimination auf nichtlineare Gleichungssysteme.
Z.B. das Gröbner-Basis-Verfahren - in Programmiersprache CAL angewendet. Kann keine trigonometrischen Funktionen behandeln. Der Nachteil von diesem und ähnlichen Verfahren sind ihre hohe Komplexität (doppelt exponentiell).

Konsistenzverfahren auf Basis der Intervallarithmetik.
Seit den sechziger Jahren in der Künstlichen Intelligenz entwickelt.Z.B. in BNR-Prolog. Vorteil polynomialer Komplexität. Als Erweiterung der Intervallbereiche von *FD* auf beliebige arithmetische Funktionen und auf reelle Zahlen verstehbar. Im Gegensatz zu *FD* aber unendlicher Wertebereich.

Anmerkung

Definition der Kantenkonsistenz aus FD muß weiter abgeschwächt werden: Nicht nur Löcher durch $\neq$ und Multiplikation zweier Variablen. Auch die Intervallgrenzen selbst müssen keine Lösung sein wegen Rundungsfehlern (die sich aufaddieren).

Intervallbereiche in I

```
durchschnitt @ X::A:B, X::C:D <=> X::max(A,C):min(B,D).
nicht_leer @   X::Min:Max <=> Min>Max | fail.
determiniert @ X::Min:Max ==> Min=Max | X=Min.
unifiziert @   X::Min:Max ==> ground(X) | X::X:X.

le @ X le Y, X::A:B, Y::C:D ==> Y::A:D, X::A:D.
eq @ X eq Y, X::A:B, Y::C:D ==> Y::A:B, X::C:D.
ne @ X ne Y, X::A:A, Y::A:A <=> fail.

add @ add(X,Y,Z), X::A:B, Y::C:D, Z::E:F ==>
                  X::E-D:F-C, Y::E-B:F-A, Z::A+C:B+D.
```

Anmerkung

Möglichst wie Intervalle in Constraintsystem *FD*, allerdings Probleme z.B. mit striktem <, da Nachfolger eine reellen Zahl nicht definierbar.

Rechnen mit den Intervallgrenzen soll wie in *FD* Kantenkonsistenz sicherstellen, allerdings muss man z.B. bei Multiplikation zweier Variablen die Zahl 0 mitberücksichtigen, falls sie im Intervall der Variablen vorkommt. Noch komplizierter ist es z.B. bei Potenzierung und trigonometrischen Funktionen. Grund: Diese Funktionen sind nicht mehr monoton (z.B. "$-$ mal $-$ ist $+$").

Multiplikation in I

```
        mitnull(A:B) :- A=<0, 0=<B.

mult_z @ mult(X,Y,Z), X::A:B, Y::C:D ==>
        M1 is A*C, M2 is A*D, M3 is B*C, M4 is B*D,
        Z::min(min(M1,M2),min(M3,M4)):max(max(M1,M2),max(M3,M4)).

mult_y @ mult(X,Y,Z), X::A:B, Z::E:F ==>
        not mitnull(A:B) |
        M1 is E/A, M2 is E/B, M3 is F/A, M4 is F/B,
        Y::min(min(M1,M2),min(M3,M4)):max(max(M1,M2),max(M3,M4)).
mult_x @ mult(Y,X,Z), X::A:B, Z::E:F ==> ...
```

Anmerkung

`mult(X,Y,Z)`, d.h. $X * Y = Z$:
Aus $X = 0$ folgt $Z = 0$ und Y beliebig.
Aus $Z = 0$ folgt $X = 0$ oder $Y = 0$.
Aus $Z \neq 0$ folgt $X \neq 0$ und $Y \neq 0$.
Aus $X \neq 0$ und $Y \neq 0$ folgt $Z \neq 0$.
Gilt analog für Intervalle von X, Y, Z.

```
A::0:0.3, B::0:0.3, C::0:0.3, A le C, mult(A,B,C).
% ergibt
A::0.0:1.4285714285714285e-07,
B::0:0.3,
C::0.0:1.4285714285714285e-07,
A le C, mult(A,B,C).

A::-3:3, B::-3:3, C::-3:3, A eq C, mult(A,B,C).
% L"osungen A=C=0 oder B=1, kann man nicht
% endlich "uber gleiche Variable enumerieren
A:: -3:3,
B:: -3:3,
C:: -3:3,
A eq C, mult(A,B,C).
```

Multiplikation in I, Fortsetzung

Der schwierige Fall:

```
mult_xyz @ mult(X,Y,Z), X::A:B, Y::C:D, Z::E:F ==>
          mitnull(A:B), mitnull(C:D), not mitnull(E:F) |
          (A*C<E -> D>0, X::E/D:B ; true),
          (B*D<E -> C<0, X::A:E/C ; true),
          (F<A*D -> C<0, X::F/C:B ; true),
          (F<B*C -> D>0, X::A:F/D ; true).
```

Anmerkung

```
mult(X,Y,Z), X:: -2:3, Y:: -3:4, Z::7:12.

mult(X,Y,Z), X:: -2:3, Y:: -3:4, Z:: -12: -9.
```

Nichtlineare Gleichungssysteme I

(1) $X^4 - 12X^3 + 47X^2 - 60X =_I 0$
(2) $X^4 - 12X^3 + 47X^2 - 60X + 24 =_I 0$
(3) $X^4 - 12X^3 + 47X^2 - 60X + 24.1 =_I 0$

Intervallpropagierung liefert für Gleichung (1) das Intervall $0.0 : 5.0$ für die Variable X, (1) hat die Lösungen $0, 3, 4$ und 5.
Für (2) erhalten wir $0.8 : 1.0$, Lösungen sind $0.888305...$ und 1.
Für (3) gibt es keine Lösung.

Es müssen also nicht einmal die Grenzen eines Intervalls Lösungen darstellen: Die Lösung $0.888305...$ ist eine irrationale Zahl und kann durch eine Fließkommazahl nur approximiert werden.

Anmerkung

Intervallpropagierung mit dem Constraint $\texttt{X}=_I\texttt{X*2}$ zum Beispiel halbiert in jedem Schritt das Intervall für `X` und dies ist (zumindest theoretisch) unendlich oft möglich. Terminierung und Effizienz erreicht man durch ein Abbruchkriterium, das eintritt, wenn das Intervall eine vorgegebene Größe unterschreitet.

Dadurch (und wegen der Ungenauigkeit der Maschinendarstellung reeller Zahlen) ist es mit Intervallverfahren aber praktisch nicht möglich, eine Variable auf genau einen Wert einzuschränken. Ein Intervall über reellen Zahlen, das nicht nur einen Wert erlaubt, beinhaltet aber immer unendlich viele Werte. I. a. läßt sich mit Intervallpropagierung allein nicht sicher sagen, ob, wo und wie viele Lösungen in einem Intervall stecken.

Erweiterungen

Intervallpropagierung ergänzen durch:
- Enumerationsverfahren
- Probing

Intervallpropagierung verbessern durch:
- Konvergenzbeschleunigung
- Gröbere Granularität der Constraints

Intervallpropagierung mit anderen Verfahren kombinieren:
- Variableneliminierung (Gröbner Basen)
- Newton'sches Näherungsverfahren...

Anmerkung

Enumerationsverfahren kann helfen, das Intervall in zwei Hälften teilt und diese getrennt untersucht. Da dies unendlich oft möglich ist, Abbruchkriterium notwendig.

Anwendungen

Anwendungen für Intervallconstraints finden sich in Physik, Geometrie, Mathematik und Chemie, vor allem aber in der Modellierung und Simulation (Simulation analoger Schaltkreise, räumliches Schließen, zum Beispiel für Roboterarmsteuerung, Gleichgewichtszustände von Chemikalien, Finanzanalysen).

Anmerkung