時相論理型言語 Tokio -ITL における Unification の考察

2X-8

河野 真治, 中村 宏, 藤田 昌宏* 田中 英彦
(東京大学 工学部, *富士通研究所)

1 はじめに

我々は、ITL に基づく時相論理型言語 Tokio についての研究を行なっている。最近は、Abadi の Templog, 京大桜川氏の Teomporal Prolog などの言語がでてきている。Tokio の目的は基本的には、Hardware Description にあり、このような分野のものとしては、同じく ITL を元にした Tempura がある。その中で Tokio は、その Unification の方法に特徴を持っている。

Hardware Description において Unification は、各種の信号線を記述するのに優れている。例えば、Bus, Signal, Register, Data Path, Analog Sw 等である。今までは、これらは、そのための専用の記述や、レジスタでの代用、命題変数等でばらばらに記述されてきた。

2 今までの Unification

この場合、Unification の対象となるのは、アトムと呼ばれる定数、変数とファンクタと呼ばれる関数型 (複数の変数または項の集合と関数の名前を表すアトムの集合) 、である。これらは、適当な関数とその値、述語を表している。 Unification とは、変数に適切な値を代入して、二つの項を変数の名前の差を除いて、等しくする作業である。このうち最も置換の少ないもの MGU (Most General Unifier) を求める事が重要である。このためには、両方の項を並行して、 depth first に比較代入していく方法がとられる。

3 LTTL 上の Unification

これに対し LTTL 時相論理では、変数の値、述語の値が時刻毎に定まる点が異なる。二つの項は、ここでは、適当な時相論理演算子を含んだ変数とファンクタだとする。ここで使われる時相論理演算子は、□ (いつも)、◇ (いつか)、○ (次の時刻) である。例えば、

$$p(\Box a(A), \bigcirc B, C)$$

の様に使う。

[0]Temporal Logic Progarmming Language Tokio: A note on Unication in Interval Temporal Logic
Shinji KONO, Hiroshi NAKAMURA, Masahiro FUJITA, Hidehiko TANAKA
University of Tokyo

4 Abadi の方法

Abadi [TTP] の方法は、時相論理演算を普通のファンクタと見て unification をしていく。さらに、

$$\bigcirc u(A0, A1 \cdots An) \equiv u(\bigcirc A0, \bigcirc A1 \cdots \bigcirc An)$$

として、○ は中に入れられるとする。

$$increment(A, s(A)).$$

$$?- increment(0, \bigcirc X), \bigcirc p(X).$$

これは、$0 = \bigcirc 0$ なので、

$$?- \bigcirc increment(0, X), \bigcirc p(X).$$

となり、

$$?- \bigcirc p(s(0)).$$

となる。

5 Tokio での方法

この方法では常に、○ の入っている項は、○ を外に出して、 (未来に送り込んで) 評価しなければならない。これは、Temporal Logic Theorem proving を Temporal Logic Programming としてみる時には、この項全体を次の時刻のために複写する事になるので望ましくない。今の例で必要なものは、$\bigcirc X = s(0)$ である。ここで、Unification の置換を各時刻において行なう事にする。つまり、時相論理の変数は、各時刻毎の値を持つのだから、変数を各時刻毎の値の集合としてみる訳である。○ は、その特定の時刻を抜き出す accesor になる。X の値の集合を $[X_1, X_2 \cdots X_n]$ とすると、$\bigcirc X$ は、X_1 をさす。この方法だと前の例は、

$$A = 0, s(A) = \bigcirc X$$

の Unification となり、$X_1 = s(A)$ が直ちに得られる。ここで、時間的に定数となるもの、□ が付いているものは、すべての時刻の値と Unification するものとする。

ここで問題となるのは、通常 LTTL がとり扱う時間は、無限なので、X の各時刻の集合も無限になってしまう事である。これは、値の集合をさらに、量化する事により解決する。これはまた、値の集合を要求駆動で生成する事に相当する。時間方向の定数は、この量化により、ただ一つの要素を持つ集合として扱う事ができる。

$$\exists Y[X_1, s(0), Y]Y \equiv [X_2, X_3 \cdots X_n]$$

実際には、これらの変数は、リスト構造を使って実現する。· を使って、

$$Y = X_2 \cdot (X_3 \cdot X_rest)$$

となる。

6 ITL 上の Unification

さらにこれを ITL (Interval Temporal Logic) に適用する。ここで、 ITL は、時相論理演算子として、○ (先頭が、一つ短くなった時間区間), empty (空の時間区間)、&& (chop) を持ち、これは、一つの時間区間を二つに分けるという意味を持つ。ITL では、変数の値、述語の真偽値ともに、各時間区間毎に定まる。

さらにここで、変数の値は、時間区間の先頭の時刻でのみ定まると仮定すると (locality)、変数の値は、各時刻毎に定まり、LTTL と同様の方法を使う事ができる。local でない関数ついては、Abadi の方法を次のように拡張する事ができる。

$$p(A\&\&B, C\&\&D) \equiv p(A, C\&\&D)\&\&p(B, C\&\&D)$$

一つの述語の中の二つの && は、違う時区間の分割を表している。各時区間に対応する値を変数に割り当てる事により、この場合も Unification を実現できる。しかし、どちらの場合も多数の時区間の分割を生成してしまう。

ここで、時間区間の最後を表す時相論理演算子 fin を考えよう。

$$fin(X) \equiv \exists Y(Y\&\&(X, empty))$$

と定義する事ができる。この値は、もちろん時間区間の最後に依存するので、 local ではない。さらに、= を現在時刻だけで等しいという意味に ITL 上で定義する。

$$eq(X, X).$$

$$X = Y \equiv \exists Z(eq(X, Y), empty\&\&Z)$$

ここで、$eq(X, Y)$ は、変数X, Y がすべての時間区間で同じ値を持つ事を表す。さらに関数としての fin を定義する。

$$fin(A) = B \equiv fin(A = B)$$

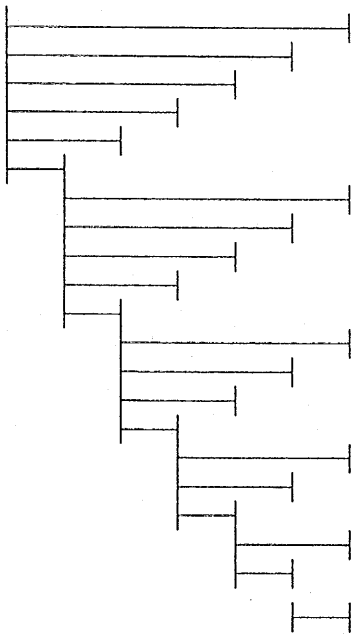
ここで、B は、時間的に定数である。

local でない変数の値は、ある変数X の $fin(X)$ を使ってすべて取り出す事ができる。local でない変数の値は、時刻x から、時刻y までの時間区間に対して定まる。ここで、各時刻x に対してある変数 X_x を割り当てる。さらに、時刻y で終了する時間区間に対する$fin(X_x)$ を時刻xから時刻y への値に割り振ればよい。

これにより、ITL 上の Unification を各時刻に割り振られた、LTTL の変数同志の Unification に帰着させる事ができる。

References

[TTP] Abadi,M., "Temporal-Logic Theorem Proving", STAN-CS-87-1151, March 1987



Several Intervals in a Interval