アクティブデータベースにおける弱合流性概念の提唱

4K－5

中山 具之† 富井 規雄¶ 土屋 隆司¶ 池田 宏†¶
電気通信大学情報システム学研究科† 鉄道総合技術研究所¶

1. はじめに

近年、従来のデータベースにルール実行機構を加えたアクティブデータベースが注目されており、分散協調作業の支援環境など様々な応用が検討されている([1])。アクティブデータベースにおいては，合流性の概念が重要である。合流性とは、最終データベース状態がルールの実行順序にかかわらず同じであるという性質のことを言う。ルール集合が合流性を満たさない場合、ルールの実行順序によってデータベースの最終状態が異なった状態になってしまう可能性がある。

しかし、実際の応用例では、データベースの最終状態が完全に一致する必要はなく、着目している変数の値のみが一致すれば十分であるという場合がある。例えば、複数のサブシステム間で共有されるグローバルな変数の値は一致している必要があるが、各サブシステムのローカルな変数については、値が異なった結果となっても差し支えはない、といった状況が考えられる。

このような性質を表現するために、本稿では，「弱合流性」という概念を提唱する。本稿では，弱合流性を、「アクティブデータベースにおいて、ルールの実行順序にかかわらず、着目する変数の値が同一であるようなデータベースの最終状態をとること」と定義する。

合流性を満たさない場合、一般にルール間に優先順位を設定することで、競合ルールの存在を減らし、合流性を満たそうとするが、本稿の弱合流性ではこの作業自体（優先順位の設定）を軽減することが可能になる。

2. 弱合流性概念

図1に示すような鉄道における、分散協調スケジューリングシステムのためのアクティブデータベースの例を考える。

ここで、列車ダイヤサブシステムは、列車の時刻と番線を決定する。乗務員運用サブシステムは乗務員の勤務内容を決定する。構内入換計画サブシステムは駅構内における車両の入換時刻を決定する。

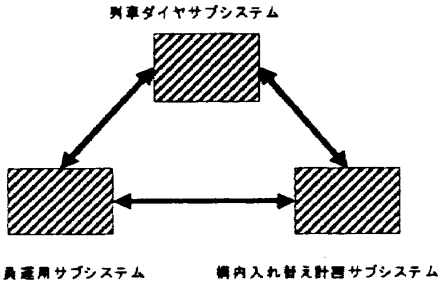

図1:分散協調スケジューリングシステム

この３つのサブシステムは互いに連携をとって、全体の輸送計画（列車ダイヤ、乗務員運用計画、構内入換計画）を作成していく。その過程では、各サブシステムの変数の値の変更が他サブシステムに伝えられ、全体としてのスケジュールの整合性が保たれる。このアクティブデータベースは、図２に示す５個のルールを持つとする。

```
rule r1 {
    Event:      update x1
    Condition:  x2 < x1 + 180
    Action:     x2 = x1 + 180
    Precedes:
    Follows:    r3, r4, r5
    }
rule r2 {
    Event:      update x1
    Condition:  y < x1 + 120
    Action:     y = x1 + 120
    Precedes:
    Follows:    r3, r4, r5
    }
rule r3 {
    Event:      update y
    Condition:  z != y + 60
    Action:     z = y + 60
    Precedes:   r1, r2, r4, r5
    Follows:
    }
rule r4 {
    Event:      update x2 or y
    Condition:  x2 < y < x2 + 60
    Action:     y = x2 + 60
    Precedes:   r1, r2
    Follows:    r3
    }
rule r5 {
    Event:      update x2 or y
    Condition:  x2 ≧ y > x2 - 60
    Action:     y = x2 - 60
    Precedes:   r1, r2
    Follows:    r3
    }
```

図2:ルール集合の例

今、図３に示すようにスケジュールが作成途上にあるときに（この図は時間の進行に伴う車両の番線占有状態を示している)、例えば、初期値{ $x1 = 0, x2 = 180, y = 120, z = 180$ }に対して $x1$ の値が 60 に変更されたとする。

このとき、２つのルール実行順序が存在しうる。一つは { $r1 \rightarrow r2 \rightarrow r3$ }でこの時の最終状態は{ $x1$

A Proposal of Weak Confluence Concept in Active Database
Tomoyuki NAKAYAMA† Norio TOMII¶ Ryuji TSUCHIYA¶ Hiroshi IKEDA†¶
The Graduate School of Information Systems, The University of Electro-Communications. †
Railway Technical Research Institute¶ e-mail : jin@rtri.or.jp

= 60, $x2$ = 240, y = 180, z = 240 }である。

もう一つは{ $r2$→ $r3$→ $r4$→ $r3$→ $r1$→ $r4$→ $r3$ }となり、この時の最終状態は{ $x1$ = 60, $x2$ = 240, y = 300, z = 360 }である。

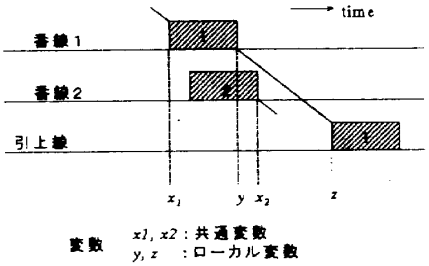

図3:作成途中のスケジュール

すなわち、最終状態が一致しないので、このルール集合は合流性を満たさない。しかしながら、着目する変数を、2 つ以上のサブシステムに共通する変数（$x1$, $x2$）とした場合、それらの変数の値は、2 つの最終状態において一致している。このような性質を「弱合流性」と呼ぶことにする。

弱合流性の判定アルゴリズム

3.1 判定アルゴリズム

一般の合流性の判定については、優先順位の付けられていないルールおよびそれらによってトリガされるルールの集合の間の可換性判定に基づくアルゴリズムが提案されている（[2]）。

このアルゴリズムの適用対象とするルールを次の手続きで選定することにより、弱合流性を判定することが可能になる。

1. すべてのルールから、着目する変数に対してアクションを実行するルールを選び出し、それを *Sig* に入れる。
2. *Sig* に入れられたルールと可換ではないルールを *Sig* に加える。この作業を繰り返す。
3. [2]のアルゴリズムを *Sig* に適用し合流性を判定する。

3.2 可換性判定条件の拡張

3.1 節で挙げたアルゴリズムで弱合流的であると判定されたルール集合は必ず弱合流的であるが、その逆は真ではない。例えば、図 2 で挙げた例は弱合流的であるが、3.1 節のアルゴリズムでは弱合流的であるとは判定されない。

その理由は、ルール *r4* とルール *r5*、この 2 つのルールに原因があり、この 2 つのルールは同じ変数にアクションを実行するので可換ではないと判断されてしまい、そのためにこのルール集合は合流的ではないと判定される。

そこで、スケジューリングシステムにおいて利用されるルール記述に特化した可換性判定を導入することによって、より現実に即した可換性判定が行えるようにする。具体的には、ルール *r4* とルール *r5* を見ると、この 2 つのルールの異なっている部分は Condition だけである。しかもアクションを実行する変数 y の範囲は重なっていないことがわかる。よってこの 2 つのルールはどちらか一方しか実行されないので実は可換であると判断できる。

スケジューリングシステムにおけるルールは数値の変更に関わるルールが主であると思われ、そのようなルールのペアが次の 2 つの性質を満たすとき、可換であると判断することを提唱する。

1. 2 つのルールが同じ変数を読み込み、同じ変数にアクションを実行する。
2. 2 つのルールが Condition 部分において Action 部分で実行される変数の範囲が重なっていない。

この条件の追加により図 2 の例のルール集合は合流的であると判定できる。

3. まとめ

分散協調作業環境を支援するアクティブデータベースに対して弱合流性概念を提唱し、その判定アルゴリズムを示した。今後より効率的に弱合流性を判定できるように更なるアルゴリズムの効率化を検討する。

参考文献

[1] 伊藤 聡, 溝口 祐美子, 吉瀬 仁志, 月本 洋: エンジニアリング環境モデルに基づく協調設計支援システム, pp.39-44, 電子情報通信学会, 信学技報 OFS96-21, AI96-13(1996-07).

[2] A.Aiken, J.Widom, and J.Hellerstein. Behavior of database production rule: Termination, confluence, and observable determinism.In Proceedings International Conference on Management of Data, pp.59-68, San Diego, CA, 1992.