

先読みスケジューラにおける宣言の正確さについて

茨木俊秀⁺, 亀田恒彦⁺⁺

⁺ 京都大学工学部数理工学科

⁺⁺ サイモンフレイザー大学計算学部

データベースの並行処理制御に用いられる先読みスケジューラCS(C)は、各トランザクションがあらかじめ宣言した読み取り集合及び書き込み集合の情報を利用して、後退復帰なしに直列可能スケジュールを出力する。宣言が正確でない場合には後退復帰を必要とすることがあるが、本報告では、新しく導入したスケジュール間の距離を用いて以下の性質が成立することを示す。

(i) 宣言が正確である場合、入力系列 s に対して出力されるスケジュール s' はC中のスケジュールのうち s に最も近いものである。

(ii) 宣言された集合が実際より大きいとき、場合aが場合bより正確ならば、出力される s^a と s^b について、 s^a は s^b より s に近い。

(iii) 宣言された集合が実際より小さいときも同様の性質が成立する。この時、場合aが後退復帰を必要とするならば、場合bでも必要である。

On the Accuracy of Predeclarations for Cautious Schedulers

Toshihide IBARAKI⁺ and Tiko KAMEDA⁺⁺

⁺ Faculty of Engineering, Kyoto University, Kyoto, Japan 606

⁺⁺ School of Computing Science, Simon Fraser University, Burnaby, B.C., Canada V5A 1S6

A cautious scheduler CS(C) for database concurrency control requires each transaction to predeclare its read and write sets. Based on this information, CS(C) outputs a serializable schedule in C without resorting to rollbacks. This paper shows the following properties, in terms of a new distance measure defined between schedules, assuming that the predeclarations may not be correct.

(i) If predeclaration is correct, the sequence s' output for an input sequence s is the schedule in C that is closest to s .

(ii) If read and write sets larger than the correct ones are predeclared, and if case a is more accurate than case b, the output schedule s^a of case a is closer to s than the output schedule s^b of case b.

(iii) Similar properties as (ii) hold for the case in which read and write sets smaller than the correct ones are predeclared. Furthermore, if case a necessitates rollback, then so does case b.

1. まえがき

データベースの並行処理制御において、従来の方式に変わり、先読みスケジューラ (cautious scheduler) CS (C) が提案されている [1, 3-7]。この方式では、各トランザクションはその実行の要求に先立ち、読み取り集合及び書き込み集合をあらかじめ宣言しておくことが要求される。その見返りとして、宣言が正確であれば、システムデッドロックを生じることなく、また後退復帰 (rollback) も必要としないという特徴を持つ。

しかし、トランザクションによっては、あらかじめその読み取り集合と書き込み集合を知ることが困難であることも考えられる。すでに文献 [3, 4] において、各トランザクションがあらかじめ大きめの読み取り集合と書き込み集合を宣言しておき、実行にともなってその一部を取り消す場合について考察し、本報告で扱ういくつかのタイプの先読みスケジューラでは、デッドロックや後退復帰を生じることではないことを示した。また [5] において、宣言が不十分である場合についても考察し、この場合にはデッドロックを回避するためにトランザクションの取り消しとそれにともなう後退復帰が必要となる場合があることを示し、その手順を与えた。また、同文献では、宣言を全く行わない場合について、遅延なく処理されるスケジュールのクラスを明らかにしている。

本報告では、文献 [5] と同様、各トランザクションが、ステップの実行要求に際して、読み取り集合と書き込み集合の宣言を変更することを許すモデルを前提とする。ただし、常に実際の読み取り集合と書き込み集合を含むものが宣言されている場合と、常に、部分集合になっている場合の二つについて考察する。なお、後者の場合、宣言されていないステップの到着も可能である。いずれの場合も、全てのステップが完了したとき、各トランザクションはその旨スケジューラに知らせるものとする。

得られた主要な結果は次の通りである。スケジュール同志の近さを示すある関係を導入することで、次の性質が言える。

1. 正確な宣言がなされたとき、入力スケジュール s に対し先読みスケジューラ CS (C) が出力するスケジュール s' は、 $s' \in C$ を満たすものの中で s に最も近いものである。

2. 宣言が余分になされる場合 a と b について、 a が b より正確であるとする、入力スケジュール s に対し出力されるそれぞれの場合のスケジュール s^a と s^b について、 s^a の方が s^b より s に近い。

3. 宣言が不足している場合の a と b について、 a が b より正確であるとする、2 と同様の性質が成立する。また、 a において後退復帰が必要であるとする b においても必要である。

このように、不完全な宣言の下での先読みスケジューラの動作は、宣言が正確になされるほど、その出力は正確な場合に近くなり、並行処理機能も向上することがわかる。

2. データベースシステム

データベースシステムはデータ項目の集合 $D = \{X, Y, \dots\}$ とトランザクションの集合 $T = \{T_0, T_1, T_2, \dots, T_r\}$ からなる。一つのトランザクション T_i はいくつかの読み取りステップ $R_i[S]$ と書き込みステップ $W_i[S']$ の系列である。ただし、 $S, S' \subseteq D$ 。読み取りステップ $R_i[S]$ とは各 $X \in S$ に対する読み取り操作 $R_i[X]$ をまとめて記したもので、 $R_i[X]$ はデータ項目 X の内容を読み取るという操作である。 $W_i[S']$ も同様に定義され、とくに $W_i[X]$ はデータ項目 X の内容を更新することを意味する。なお $T_0 = W_0[D]$ と $T_r = R_r[D]$ は特別なトランザクションであって、 T_0 は他のトランザクションの実行に先立ち、全てのデータ項目の初期値を準備し、 T_r は全てのトランザクションの後で、残されたデータ項目を読み取る。

データベースシステムに入力されたトランザクションのステップを一行に並べた系列をスケジュールという。たとえば

$$\begin{aligned}s = & W_0[D] R_1[X] W_1[Y] \\ & W_2[X, Y] R_3[Y] W_1[X] \\ & W_3[X] R_r[D]\end{aligned}$$

である。各読み取り操作 $R_j[X]$ はそれに先立ち、しかも最も新しい X への書き込み操作 $W_i[X]$ の結果を読み取る。例えば、上の例で $R_3[Y]$ は $W_2[Y]$ の結果を読み取る。(厳密にはこれは単版モデルであり、多版モデルでは $([4, 10])$ など) $R_j[X]$ に先立つ $W_i[X]$ の中から読み取るべき版を選択できる。)

並行処理スケジューラの役割は、入力されたスケジュール s を、必要ならば変更して、直列可能 (serializable) なスケジュール s' を得ることにある。直列可能とは、各トランザクションごとにその全てのステップを、他のトランザクションのステップと混合することなしにまとめて実行していくという直列スケジュールと、実質的に同一と見なせることを意味し、データの首尾一貫性を保証する十分条件となっている (詳しくは [2, 9, 10] 等参照のこと)。直列可能なスケジュールの集合を SR と記すと、 SR は全てのスケジュールの集合の真部分集合である。また、あるスケジュールが直列可能かどうか、つまり $s \in SR$ かどうかの判定は NP 完全であることが知られており [9]、従って、判定が多項式時間で可能であるような SR の部分クラスの研究が進められてきた。その様なクラスとして、本報告で用いる WW , WRW , MWW , $MWRW$ の4種がある (定義は [3, 4])。

3. 先読みスケジューラ

C を直列可能なスケジュールの一つのクラスとする ($C = WW$ 等)。各トランザクションはあらかじめ読み取り集合と書き込み集合を宣言し、その後ステップの実行を要求する。これらのステップは発生順序にスケジューラに入力され、スケジューラは最終的に C に属するスケジュールが得られるようにそれらの順序を調整しながら出力する (実行を許可する)。 P をある時点ですでに出力された部分スケジュール、 q を判定すべきステップとし、更に

DEL : その時点で実行を遅延されているステップのリスト、

$PEND = (\text{スケジューラに到着しているトランザクションの読み取り集合及び書き込み集合から定まる操作の集合}) - (Pq \text{ 中のステップを構成する操作の集合})$

とする。一般に、 $DEL \subseteq PEND$ である。各トランザクションはその前のステップの実行が完了してはじめて次のステップを発するから、 DEL には各トランザクションについて、たかだか1個のステップが含まれている。また $T_0 = W_0[D]$ は常に P の先頭に位置する。

クラス C に基づく先読みスケジューラ $CS(C)$ は、上の状況において、部分スケジュール Pq に

$PEND$ 中の操作を適当な順序で接続すれば (一番最後に $R_i[D]$ が来る)、 C に属するスケジュールが得られるかどうかを判定する。これを完成テスト (completion test) という。完成テストが成功すると $CS(C)$ は q の実行を許可し、失敗ならば q を DEL に加える。また、成功の場合、 DEL 中のステップで実行可能になるものがないかをさらに調べる。 DEL 中のステップの順序は到着順とする。

先読みスケジューラ $CS(C)$ の手順

$CS0$ (初期化): $P := W_0[D]$, $q :=$ (最初に到達したトランザクション T_1 の最初のステップ)、 $DEL := \Lambda$ (空きリスト)、 $PEND := (T_1 \text{ が宣言した操作の集合})$ 。

$CS1$ (完成テスト): $PEND := PEND - \{q\}$ とし Pq に対する完成テストを加える。テストが失敗すれば $CS2$ へ、成功すれば q の実行を許可し $CS3$ へ進む。

$CS2$ (q の実行の延期): $PEND := PEND \cup \{q\}$ とし、 q の性質に応じて以下の1つを実行する。

(a) $q \in DEL$ かつ DEL の全てのステップの完成テストが失敗した場合: $CS5$ へ進む

(b) $q \in DEL$ であるが (a) ではない: DEL の次のステップを q とみなし $CS1$ へ戻る。

(c) $q \notin DEL$: q を DEL の最後尾に加え、 $CS5$ へ進む。

$CS3$ (q の実行): $P := Pq$ とする。 $q \in DEL$ ならば DEL から q を除く。

$CS4$ (DEL のテスト): $DEL \neq \Lambda$ ならば DEL の次のステップ (すなわち、まだテストされていない先頭のステップ) を q とみなし、 $CS1$ へ戻る。 $DEL = \Lambda$ ならば $CS5$ へ進む。

$CS5$ (次の到達ステップ): q を次に到達するステップとする。もし q があるトランザクション T_i の最初のステップならば、 $PEND := PEND \cup (T_i \text{ が宣言した操作の集合})$ と置く。 $CS1$ へ戻る。□

4. 完成テスト

先読みスケジューラの核は $CS1$ の完成テストにある。ここでは、 $CS(WW)$, $\overline{CS}(WRW)$, $CS(MWW)$ 及び $CS(MWRW)$ の4種について、その実行方法を簡単に述べる [3, 4]。なお、 $\overline{CS}(WRW)$ は $CS(WRW)$ に若干制限を加え

たもので、多項式時間での実行を保証するために導入された[3]。またMWWとMWRWのMは多版モデルであることを意味する。

与えられたPqとPENDに対し、活性TIOグラフ $G = ATIO(Pq, PEND)$ を以下のように定義する。Gはすでに到着しているトランザクションを節点として持ち、部分スケジュールPqにおいて $R_j[X]$ が $W_i[X]$ の結果を読み取るとき、節点 T_i から節点 T_j へのラベルX付きの有向枝 $(T_i, T_j) : X$ をもつ。これを読み取り枝という。更に、ある書き込み操作 $W_i[Y]$ から読み取るようなPq中のステップが存在しないとき、 $W_i[Y]$ は不用であるといい、追加節点 T'_i を付し、有向枝 $(T_i, T'_i) : Y$ を加える。PEND中の $W_i[X]$ も追加節点 T'_i を用いて、有向枝 $(T_i, T'_i) : X$ で示される。PEND中の $R_i[X]$ は頭部を持たない T_i への矢印(ラベルX)で示されるが、後述の排除枝の計算においては無視される。

更に次の制約枝を定義する。

ww-枝: ある $X \in D$ に対し、Pq中において $W_i[X]$ が $W_j[X]$ に先行するかあるいはPq中に $W_i[X]$ があり、かつ $W_j[X] \in PEND$ の時、 (T_i, T_j) をww-枝という。

wr-枝及びrw-枝も同様に定義される。これらに基づいて、各クラスCの活性TIOグラフ $ATIO_C(Pq, PEND)$ を次のように定義する。

$CS(WW) : G = ATIO(Pq, PEND)$ に ww-枝を加える。更に各 $X \in D$ に対し、Pq中の最後の書き込み操作を $W_k[X]$ とすると、 $R_j[X] \in PEND$ ならば (T_k, T_j) を加える。

$\overline{CS}(WRW) : wr$ -枝及び rw -枝を加える。更に、上記の $W_k[X]$ に対しP中に $W_i[X]$ ($i \neq k$)があれば (T_i, T_k) 、更に $W_j[X] \in PEND$ ならば (T_k, T_j) を加える。

$CS(MWW) : ww$ -枝を加える。更に $q = R_i[S]$ ならば、 $W_h[X] \in PEND$, $X \in S$, に対し (T_i, T_h) を加える。

$CS(MWRW) : wr$ -枝及び rw -枝を加える。

なお、 $q = R_i[S]$ の場合、 $CS(WW)$ 及び $\overline{CS}(WRW)$ では、各 $X \in S$ に対し、P中最後の

$W_k[X]$ に基づいて読み取り枝 $(T_k, T_i) : X$ が書かれる。これは単版モデルでは必ず最新の版を読むからである。しかし、多版モデルの $CS(MWW)$ と $CS(MWRW)$ ではそのような枝は加えられない。完成テストの手順の中でどの版から読むかの決定がなされる。

次に、上記の有向グラフに次の枝を導入する。

排除枝: 同じラベルXを持つ $(T_h, T_i) : X$ と $(T_j, T_k) : X$ において、 $h \neq j$ かつ T_h から T_k への路が存在するならば、枝 (T_i, T_j) を付す。

可能な排除枝をすべて加えた結果を排除閉包(exclusion closure)という。

定理4.1 [3, 4] 先読みスケジューラ $CS(WW)$, $\overline{CS}(WRW)$, $CS(MWW)$, $CS(MWRW)$ において、それぞれ上記のように活性TIOグラフの排除閉包を定義する。この時、完成テストが成功するための必要十分条件は、排除閉包が閉路を持たないことである。□

閉路の存在の判定は多項式時間で可能である。

例4.1 $CS(WW)$ において次のPqおよびPENDに対する活性TIOグラフの排除閉包は図1である。

$$Pq = W_0[D] R_1[Y] R_2[X] W_2[X] \\ R_3[X] W_3[Y] W_2[Y],$$

$$PEND = \{W_1[Y]\}$$

このグラフは有向閉路を含むので、この場合完成テストは失敗する。□

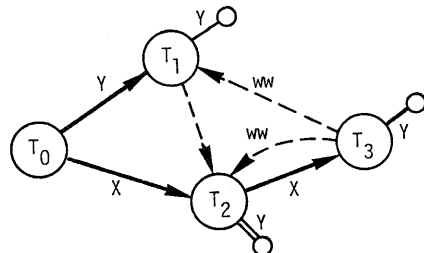


図1 例4.1の排除閉包(P中の $W_i[X]$ に対応する枝は太く、PEND中の $W_i[X]$ に対応する枝は細く、また、 $q = W_2[Y]$ は2重線で示している。)

5. 不完全な宣言の下での先読みスケジューラ

トランザクションの動作に関する仮定を緩めて、各トランザクションは、最初、あるいはステップの実行の時、読み取り集合及び書き込み集合の追加あるいは取り消しを自由に行えるものとする。宣言されていなかったステップの到着も許す。しかし、全てのステップが終了したとき、そのトランザクションはその旨スケジューラに知らせなければならない。

この前提の下で先読みスケジューラを実行するには、読み取り集合及び書き込み集合の最新の情報に基づいてPENDを更新しなければならない。その結果活性TIOグラフの排除閉包にPENDの変更にともなう閉路が生じることがある。 $G^*(Pq, PEND)$ をある活性TIOグラフの排除閉包とする。この時判定中のステップ q をPENDに移した $G^*(P, PEND \cup \{q\})$ を考え、ここに閉路が存在すれば、固定(permanent)閉路であるという。

G^* が固定閉路を持つとき、現在PENDに属している操作をどのような順序で実行しても、この閉路を除くことはできない(新しい枝が加わるだけだから)。従って閉路を構成するトランザクションをどれか取り消し(abort)、それが書き込んだデータ項目の値をその前の古い値に後退復帰(rollback)することが要求される。この時、現在の q をステップとして持つトランザクション T_k がまず取り消されるが、それにともなう作業は文献[5]に述べられている。

6. 先読みスケジューラの出力するスケジュール

同じステップ集合から得られるスケジュール s, s', s'' を考えよう。

$$s = W_0[D]S_1S_2 \dots S_m R_f[D]$$

$$s' = W_0[D]S'_1S'_2 \dots S'_m R_f[D]$$

$$s'' = W_0[D]S''_1S''_2 \dots S''_m R_f[D]$$

次の性質をもつ i が存在するとき、 $s' < s''$ と定義する。(つまり、 s' は s'' にくらべより s に近い)。

$$S'_1S'_2 \dots S'_i = S''_1S''_2 \dots S''_i$$

$$S'_{i+1} \neq S''_{i+1}$$

$$S'_{i+1} = S_p, S''_{i+1} = S_r \text{ かつ } p < r$$

さらに $s' = s''$ あるいは $s' < s''$ のとき、 $s' \leq s''$ と記す。

補題6. 1 関係 $\leq$ は S と同じステップ集合を

持つ全てのスケジュール上の全順序関係である。また、関係 $\leq$ において、 s 自身は最小のスケジュールであり、 s の逆 $s^R = W_0[D]S_mS_{m-1} \dots S_1R_f[D]$ が最大のスケジュールである。□

$WW^*, \overline{WRW}, MWW^*, MWRW^*$ を、それぞれ、先読みスケジューラ $CS(WW), \overline{CS}(WRW), CS(MWW), CS(MWRW)$ において、遅延なく出力されるようなスケジュールの集合と定義する[3, 4]。すなわち、例えば $s \in WW^*$ を $CS(WW)$ に入力すると、 s のどのステップもDELに置かれることはなく s がそのまま出力される。他の場合も同様である。

一般的に、 $s \in WW^*$ の場合も含めて次の性質が成り立つ[3, 4, 5]。

補題6. 2 スケジュール s が $CS(WW)(\overline{CS}(WRW), CS(MWW), CS(MWRW))$

に入力されたときの出力スケジュールを s' とすると $s' \in WW^*(\overline{WRW}, MWW^*, MWRW^*)$ が成立する。ただし、各トランザクションの宣言は正確であるとする。□

すでに知られているように

$$WW^* = WW$$

$$\overline{WRW} \subset \overline{WRW},$$

$$MWW^* \subset MWW,$$

$$MWRW^* \subset MWRW$$

が成り立つ。補題6. 2の s と s' は、更に次の性質を持つ。

定理6. 1 s と s' を補題6. 2のように定義する。

この時、任意の $s'' \in WW^*(\overline{WRW}, MWW^*, MWRW^*)$ に対し

$$s' \leq s''$$

が成立する。

略証 $CS(WW)$ において、 $s' \neq s''$ かつ

$s'' < s'$ とする。すなわち、ある i に対し

$$S'_1S'_2 \dots S'_i = S''_1S''_2 \dots S''_i$$

$$S'_{i+1} = S_p, S''_{i+1} = S_r, r < p$$

が成立する。このことは、 $CS(WW)$ が入力 s を処理するときに、 $P_q = W_0[D]S'_1 \dots S'_i$

S_p 以前に $P_q = W_0 [D] S'_1 \dots S'_i S_r$ の完成テストを試みたことを意味する。しかし、そうならば $s'' \in WW^*$ の仮定より s の処理において、 $q = S_r$ の完成テストは成功しているはずであり（文献[5]の補題7.1参照）、 $r < p$ の仮定は矛盾に導く。

他の先読みスケジューラについても同様である。□

7. 余分な宣言の下での先読みスケジューラ

先読みスケジューラに対し、スケジュール

$$s = W_0 [D] S_1 S_2 \dots S_m R_r [D]$$

を入力する。まず、宣言が正しく行われている場合を想定し、 $P_q = W_0 [D] S_1 \dots S_i$ の処理時の PEND を $PEND_i$ と記す。これに対し、実際の読み取り集合及び書き込み集合より余分な宣言をあらかじめ行い、その後、ステップの実行の時、徐々に宣言集合を収縮していくという状況を考える。その様な場合の2つの a と b について、上と同様に $PEND^a_i$ と $PEND^b_i$ を定義し、さらに、

$$PEND_i \subseteq PEND^a_i \subseteq PEND^b_i,$$

$$i = 1, 2, \dots, m$$

を仮定する。すなわち、場合 a は場合 b に比べ、より正確な宣言を行っている。

定理 7. 1 上の定理において、スケジュール s が $CS(WW)$ ($\overline{CS}(WRW)$), $CS(MWW)$, $CS(MWRW)$) に入力されたとし、場合 a と b の出力スケジュールをそれぞれ s^a と s^b と記す。この時

$$s' \preceq s^a \preceq s^b$$

が成立する。ただし、 s' は正確な宣言の下での出力スケジュールである。

略証 $s^a \neq s^b$ と仮定し、

$$S^a_1 S^a_2 \dots S^a_i$$

$$= S^b_1 S^b_2 \dots S^b_i$$

$$S^a_{i+1} \neq S^b_{i+1}$$

とする。先読みスケジューラが $P_q = W_0 [D] S^b_1 \dots S^b_i S^b_{i+1}$ の完成テストをおこなったとき、場合 b の活性TIOグラフの排除閉包 G^b は閉路を持たない (s^b を出力しているから)。ところで、仮定より、

$$PEND^a_{i+1} \subseteq PEND^b_{i+1}$$

であるから、同じ P_q に対する場合 a の G^a を考え

ると、 G^a と G^b は同じ節点集合を持ち、かつ、 G^a の枝は必ず G^b にも存在する。このことより G^a も閉路を持たず、完成テストは成功する。従って、 $S^a_{i+1} \neq S^b_{i+1}$ の仮定より、場合 a での $q = S^a_{i+1}$ 完成テストは、 S^b_{i+1} を試みる以前に成功していたことになり、 $s^a \preceq s^b$ である。□

この定理は、宣言が正確であるほど、出力スケジュールが s' に近い（つまり、定理6. 1によって入力スケジュール s に近い）、すなわち、並行処理能力が高いことを示している。

8. 不十分な宣言の下での先読みスケジューラ

前節と同様、不完全な宣言がなされるとするが、ここでは真の読み取り集合と書き込み集合の部分集合が宣言されている場合を考える。トランザクションの各ステップの到着に際してこれらを修正したり、また、宣言されていないステップの到着も許す。

補題 8. 1 先読みスケジューラ $CS(WW)$

(あるいは $CS(MWW)$, $CS(MWRW)$) を考え、入力スケジュール

$$s = W_0 [D] S_1 S_2 \dots S_m R_r [D]$$

に対し、正しい宣言の下での出力スケジュールを s' 、不十分な宣言の下での出力スケジュールを s'' とする。このとき、 $s' \neq s''$ ならば、不十分な宣言の下での先読みスケジューラは必ずあるトランザクションを取り消し、後退復帰を行わなければならない。

注 $\overline{CS}(WRW)$ に対してはこの補題は成立しない。

略証 $s' \neq s''$ より

$$S'_1 S'_2 \dots S'_i = S''_1 S''_2 \dots S''_i$$

$$S'_{i+1} \neq S''_{i+1}$$

なる i が存在する。正確な宣言（不十分な宣言）の下で $P_q = S'_1 S'_2 \dots S'_i S'_{i+1}$ の完成テストを行う際の PEND を $PEND'_{i+1}$ ($PEND''_{i+1}$) と記せば、仮定より

$$PEND''_{i+1} \subseteq PEND'_{i+1}$$

である。従って、定理6. 1の証明と同様、正確な宣言の下での活性TIOグラフの排除閉包 G^* に閉路が存在しないことから、不十分な宣言の下でも閉路は

存在しない。このことと $S'_{i+1} \neq S''_{i+1}$ より、
 S''_{i+1} は DEL において S'_{i+1} の前に位置しており、しかも、正確な宣言の下で $Pq = S'_1 S'_2 \dots$
 $S'_1 S''_{i+1}$ を調べたとき、 G^* に閉路が存在し、完成テストは失敗したことがわかる。この時の PEND を $PEND_{i+1}$ と記せば、不十分な宣言の下でも、 $PEND_{i+1}$ 中の全ての操作はいずれ宣言されるか到着する。活性 TIO グラフの作り方より、その時点の G^* には閉路が存在し、しかも、5 節に言う固定閉路であることを示すことができる ($\overline{CS} (WRW)$ についてはこの性質は成立しない)。したがって、その原因となったトランザクションを取り消さねば、システムデッドロックを回避できない。□

例 8. 1 入力スケジュールとして

$$s = W_0[X] R_1[X] R_2[X] W_1[X] \\ W_2[X] R_f[X]$$

を考え、正確な宣言が行われる場合と、何の宣言も行わず、各ステップが突然到着する場合を考える。前者の場合、図 2 (a) に示すように、 $q = R_2[X]$ における活性 TIO グラフの排除閉包 G^* に閉路が存在し、 $R_2[X]$ は DEL に入れられる。結局この場合の出力スケジュールは

$$s' = W_0[X] R_1[X] W_1[X] R_2[X] \\ W_2[X] R_f[X]$$

となる。

一方、宣言が行われない場合は、図 2 (a) において、 T_1 と T_2 の書き込みによる枝が存在しないので、閉路は構成されず、 $q = R_2[X]$ は実行される。そのあと、 $W_2[X]$ が到着したとき、図 2 (b) の G^* が得られ固定閉路を持つ。したがって、 T_2 の取り消しが必要となる。□

入力スケジュール

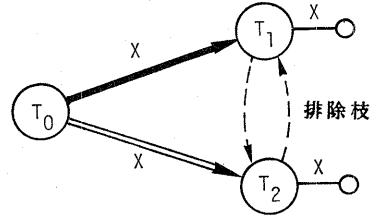
$$s = W_0[D] S_1 S_2 \dots S_m R_f[D]$$

に対し、 $Pq = W_0[D] S_1 S_2 \dots S_i$ のとき、正確な宣言の下での PEND を $PEND_i$ と記す。また、不十分な宣言を行う場合の 2 つ a と b を考え、それぞれの PEND を $PEND^a_i$ 、 $PEND^b_i$ と記し、さらに

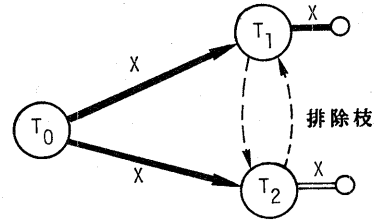
$$PEND^b_i \subseteq PEND^a_i \subseteq PEND_i,$$

$$i = 1, 2, \dots, m$$

を仮定する。すなわち、場合 a は場合 b より正確である。また、s を入力したとき、先読みスケジュー



(a) 正しい宣言の下での $q = R_2[X]$ に対する G^*



(b) 宣言のない場合の $q = W_2[X]$ に対する G^*

図 2 例 8. 1 における G^* (太い枝は実行済みの操作、二重枝はテスト中の操作、細い枝は PEND 内の操作を示す)。

ラの出力スケジュールを

正確な宣言: $s' = W_0[D] S'_1 S'_2 \dots S'_m R_f[D]$

場合 a: $s^a = W_0[D] S^a_1 S^a_2 \dots S^a_m R_f[D]$

場合 b: $s^b = W_0[D] S^b_1 S^b_2 \dots S^b_m R_f[D]$

と記そう。さらに

$$S^a_1 S^a_2 \dots S^a_{i(a)} = S'_1 S'_2 \dots S'_{i(a)}$$

$$S^a_{i(a)+1} \neq S'_{i(a)+1}$$

$$S^b_1 S^b_2 \dots S^b_{i(b)} = S'_1 S'_2 \dots S'_{i(b)}$$

$$S^b_{i(b)+1} \neq S'_{i(b)+1}$$

とする。ただし、 $i(a) = m$ あるいは $i(b) = m$ も可とする。

定理 8. 1 先読みスケジューラ $\overline{CS} (WW)$ (あるいは $\overline{CS} (MWW)$, $\overline{CS} (MWRW)$) に対し、 s^a 、 s^b および $i(a)$ 、 $i(b)$ 等を上のように定めるとき、

$$i(a) \geq i(b)$$

が成立する。

略証 $i(a)$ の定義より、場合 a での先読みスケジューラは、 $Pq = W_0[D] S^a_1 S^a_2 \dots S^a_{i(a)}$

$S'_{i(a)+1}$ 以前に $Pq' = W_0[D]S^a_1S^a_2 \dots$
 $S^a_{i(a)}S^a_{i(a)+1}$ を調べているはずである（正確な
 宣言の下で Pq が完成テストを成功させていること
 は、場合 a でも成功させることを意味するので
 $S'_{i(a)+1} \neq S^a_{i(a)+1}$ より示される）。つまり、
 DELにおいて、 $S^a_{i(a)+1}$ は $S'_{i(a)+1}$ に先行してい
 た。しかし、そうならば、場合 b でも、 Pq' は
 Pq 以前にテストされ、やはり完成テストは成功し
 ているはずである（PENDに関する包含関係から）。
 これは $i(b) \leq i(a)$ を意味する。□

すなわち、不十分な宣言の場合でも、宣言が正確
 であるほど、出力スケジューラは正確な宣言の場合
 に近くなる。補題 8. 1 の結果により、このことは、
 宣言が正確であるほど、後退復帰の可能性が小さく
 なることを示している。

なお、 $\overline{CS}$ (WRW) において、本節の結果が成
 立しないのは、活性グラフの構成に際し、テスト中
 のステップ q に関係する制約枝が導入されるが、 q
 の処理後、その様な枝が消滅してしまうことによる。
 このため PEND に関する包含関係が成立しても、活性
 TI0 グラフの枝集合の包含関係が必ずしも言えないか
 らである。実際、文献 [5] では $\overline{CS}$ (WRW) に
 対し、宣言を全く行わない方が、かえって入力スケ
 ジューラをそのまま出力できる場合があり得るこ
 とを述べている。

むすび

文献 [5] では、宣言が正確でない場合も含めて
 先読みスケジューラの構成法を述べた。本報告の結
 果は、宣言が完全に正確でなくとも、それに近けれ
 ば近いほど、その挙動が正確な宣言の場合に近づく
 ことを示している。したがって、利用者は、読み取
 り集合と書き込み集合のできるだけ正確な宣言を行
 うことが大切である。

謝辞

熱心にご討論いただいた神戸商大加藤直樹氏に深
 謝いたします。なお、本報告は一部文部省科学研
 究費およびカナダ国 NSERC の研究費によるもので
 ある。

文献

- [1] M.A.Casanova and P.A. Bernstein, General purpose schedulers for database systems, Acta Informatica 14, 195-220, 1980.
- [2] K.P.Eswaran, J.N.Gray, R.A.Lorie and I.L.Traiger, The notions of consistency and predicate locks in a database system, Comm. ACM 19, 624-633, 1976.
- [3] T.Ibaraki, T.Kameda and N.Katoh, Cautious transaction schedulers for database concurrency control, to appear in IEEE Trans. on Software Engineering; 茨木他, Cautious scheduler (先読みスケジューラ) による並行処理制御、情報処理学会データベースシステム研究会, 48-3, 1985.
- [4] T.Ibaraki, T.Kameda and N.Katoh, Multiversion cautious schedulers for database concurrency control, Simon Fraser University, LCCR TR86-5, 1986; 加藤他, 情報処理学会データベースシステム研究会, 54-5, 1986.
- [5] 茨木俊秀, T.Kameda, 不完全な宣言の下での先読みスケジューラ、電子情報通信学会、データ工学研究会, 1987年10月.
- [6] N.Katoh, T.Ibaraki and T.Kameda, Cautious transaction schedulers with admission control, ACM Trans. Database Systems 10, 205-229, 1985.
- [7] N.Katoh, T.Kameda and T.Ibaraki, A cautious scheduler for multi-step transactions, Algorithmica 2, 1-26, 1987.
- [8] H.T. Kung and C.H. Papadimitriou, An optimality theory of concurrency control for databases, Proc. ACM-SIGMOD Int. Conference on Management of Data, Boston, 116-126, 1979.
- [9] C.H. Papadimitriou, The serializability of concurrent database updates, J.ACM 26, 631-653, 1979.
- [10] C.H. Papadimitriou and P.C. Kanellakis, On concurrency control by multiple versions, ACM Trans. Database Systems 9, 89-99, 1984.