データベースシステム 116－84
（1998．7．10）

一貫性判定と処理単位の構造による協調型データベースの並行処理制御

徐　海燕[†]　古川哲也[††]　史　一華[†††]

一貫性情報の利用が，協調型データベースにおけるスケジュールの正当性に与える影響について検討する．一貫性情報を利用できれば，処理単位のACID特性を形式的に定義でき，処理単位や部分処理単位の操作結果に関する一貫性は多項式時間で判定できることを示す．部分処理単位の一貫性を用いることで検索されるデータの一貫性はスケジュールの直列可能性に依存する必要がなくなり，直列可能スケジュールのクラスを拡張した分割判定独立化可能スケジュールのクラスを導入できる．さらに，部分処理単位の数と処理単位の構造間の関係について考察することによって，分割判定独立化可能クラスの性質を分析する．

Cooperative Database Concurrency Control with Consistency Verification and Transactions' Structure

HAIYAN XU,[†] TETUSYA FURUKAWA[††] and YIHUA SHI[†††]

In this paper, we discuss the effect on correctness criterion of schedules by consistency information. Since the ACIDity of transactions are defined by using the consistency information, the consistency of results of both transactions and subtransactions can be verified in polynomial time. Consistency of read data no longer depends on the serializability of schedules because of the consistency of subtransactions, which allows us to introduce a new class of schedules, equivalent-independence, extended from the class of serializable schedules. We also discuss the property of the new class using the number of subtransactions and the structure of transactions.

1. はじめに

協同作業時に，共有データの干渉により多くの問題が生じる．データベースの応用分野がCAD/CAMなどに広がるにつれ，並行処理制御機能を有する協同作業支援機構をデータベースの機能として実現する必要性が高まってきている．データベースの並行処理制御の目的は，処理単位がACID特性と呼ばれる原子性，整合性，隔離性，耐久性の4つの性質を満たすようにすることである．

処理単位の整合性は利用者の責任において，隔離性と並行実行の整合性は直列可能性基準に基づく並行処理制御方式によって，原子性と耐久性はデータベースシステムによって保証されてきた[2]．しかし，このような並行処理制御は，協調型データベースにおいては，満足できる並行実行を提供できない[1]．この問題は広く認識されており，様々な研究がなされている[1),3)~5)]．論文4)では，作業データ間の一貫性や作業順序をデータベースの一貫性制約として利用する角度から検討している．データベースの一貫性が判断できるような一貫性制約が記述できれば，処理単位の整合性と隔離性を結果一貫性と検索一貫性として形式的に表現でき，直列可能性に依存する必要がなくなる．それに基づいて独立化可能という新たな並行処理制御の正当性基準が提案されている．

独立化可能性に基づいて並行処理制御を行うためには，処理単位に対する一貫性制御とスケジュールに対する並行処理制御が必要となる．論文4)，5)では後者の問題について具体的に検討されている．独立化可

† 福岡工業大学
Fukuoka Institute of Technology
†† 九州大学
Kyushu University
††† 福岡工業短期大学
Fukuoka Junior College of Technology

能性のためのグラフ判定方式によって許可されるスケジュールを論理可能とすると，論理可能クラスは直列可能クラスより検索操作に対する制限が緩和されているという結論が得られている．しかし，一貫性制御に関する問題はまだ議論されていない．

本論文では，一貫性情報の利用が並行実行の正当性に与える影響について考察する．まず，結果一貫性を満たすかどうかは多項式時間で判定できることを示す．それにより部分処理単位の結果一貫性も多項式時間で判定可能になる．結果一貫性を満たす処理単位からなる並行実行に対して，並行実行の整合性を保証するためには，直列可能性を必要とせず，論理可能性で十分である[4]．このため，分割判定を満たす変更結果を検索する方法で，検索一貫性が分割検索一貫性に拡張でき，それに伴って独立化可能性も拡張できる．

与えられたスケジュールが拡張された分割判定独立化可能であるかどうかは，分割数と密接に関連する．すべての処理単位が分割できなければ，直列可能スケジュールと同じ条件となる．そのため，分割数と処理単位の構造の関連を検討し，整構造というそのための処理単位構造を提案する．結果一貫性を満たす整構造である処理単位からなるスケジュールは，論理可能であれば，分割判定独立化可能となる．論理可能クラスは，直列可能クラスに比べ検索操作に対する制限が緩和されているため，多くの直列可能でないスケジュールが分割判定独立化可能となる．

本論文は，次のように構成される．2章では協同作業時の作業手順やデータ間の一貫性を一貫性制約として抽出できることを示す．3章では一貫性制約に基づくスケジュールの正当性について，4章では，一貫性情報が利用できることが処理単位のACID特性に与える影響について考察する．5章では処理単位の構造より分割判定独立化可能スケジュールの特徴を分析する．6章は全体のまとめである．

2. データベースの一貫性制約

協同作業においては，異なる作業段階で多くの作業結果が生成され，それらはさまざまなテストを受けなければならないし，複数種類の制約条件を満たさなければならない．そのような作業用データを対象とするデータベースにおける並行処理制御について議論するため，データの作業情報とデータ間の関連のみに注目し，データベースを次のように定義する．

定義1 データベースはオブジェクトの集合 O とオブジェクト間の関連の集合 R からなり，$DB(O,R)$ で表す．

- オブジェクト $o(\in O)$ は，データの値である実体部分 $e(o)$，論理値を値とする状態部分 $at_1(o)$, $\cdots$, $at_m(o)$ と論理値を値とするフラグ部分 $fg_1(o), \cdots, fg_m(o)$ からなる．
- オブジェクト o_i, o_j 間の種類 r の関連を $r(o_i, o_j)$ で表す．R は $r(o_i, o_j)$ の集合である． □

オブジェクトの実体は，論理回路やプログラムなどデータそのものである．状態 $at_k(o)(1 \leq k \leq m)$ は，オブジェクト o が k 番目のテストに合格しているまたは k 番目の制約を満足している時にのみ真である．フラグ $fg_k(o)(1 \leq k \leq m)$ は，$e(o)$ が変更された後，k 番目のテストまたは k 番目の制約の検証を受けていない場合にのみ真である．

本論文では議論を簡単にするためにオブジェクト間の関連を2項関連としているが，本論文の議論は容易に n 項関連に拡張できる．

データベースに対する操作単位は $e(o)$，$at_k(o)$ および $fg_k(o)$ であり，これらをデータ項目という．データベース中のすべてのデータ項目の集合を $\mathcal{DB} = \{e(o) \mid o \in O\} \cup \{at_k(o), fg_k(o) \mid o \in O, 1 \leq k \leq m\}$ とする．また，$I(d)$ でデータ項目 d の値を表し，データ項目集合 $D(\subseteq \mathcal{DB})$ に対して，$I(D) = \{\langle d, I(d)\rangle \mid d \in D\}$ とする．

例1 $O_1 = \{o_1, o_2, o_3\}$ であるデータベース DB_1 において，$at_1(o)$，$at_2(o)$ は回路図である実体 $e(o)$ $(o \in O_1)$ がそれぞれネットリスト抽出とシミュレーションに成功しているかどうかを記述しているものとする．x が y の部品であることを表す関連 $comp(x,y)$ を用いると，o_2，o_3 が o_1 の部品であることは，$R_1 = \{comp(o_2, o_1),\ comp(o_3, o_1)\}$ として記述できる．$\mathcal{DB}_1$ は次のようになる．

$$\left\{\begin{array}{l} e(o_1),\ e(o_2),\ e(o_3), \\ at_1(o_1),\ fg_1(o_1),\ at_2(o_1),\ fg_2(o_1), \\ at_1(o_2),\ fg_1(o_2),\ at_2(o_2),\ fg_2(o_2), \\ at_1(o_3),\ fg_1(o_3),\ at_2(o_3),\ fg_2(o_3) \end{array}\right\}$$ □

作業プロジェクトは，通常複数個の部分作業に分け

て行われる．これらの部分作業は並行に行えるものもあれば，一定の順序に従って行わなければならないものもある．複雑な作業環境では，状態の組み合せや作業の過程にはいくつかの選択枝がある場合もある．また，オブジェクトの状態は実体を正しく表していなければならない．一階述語論理の節を用いてこれらの制約をデータベースの一貫性制約として次のように形式的に記述する．

定義 2 データベース $DB\langle O, R\rangle$ の一貫性制約 $\boldsymbol{C}$ は，次のような節の集合である．

$$A_1, \cdots, A_p \leftarrow B_1, \cdots, B_q$$

ここで，$A_1, \cdots, A_p$ は状態述語 $at_k(x)$ $(1 \leq k \leq m)$ であり，$B_1, \cdots, B_q$ $(p+q \geq 1)$ は状態述語 $at_k(x)$，フラグ述語 $fg_k(x)$ $(1 \leq k \leq m)$ または関連述語 $r(x, y)$ である．x, y, z は O の要素または O 上の全称束縛変数である． □

例 2 例 1において，$at_k(x)(1 \leq k \leq 2)$ は $e(x)$ の状態を正しく表しているという制約は，$e(x)$ の変更後は新たに k番目のテストを行うまで $at_k(x)$ は真とはならないので，「$\leftarrow at_k(x), fg_k(x).(1 \leq k \leq 2)$」で表すことができる．$x$ はネットリスト抽出の後でシミュレーションに合格しなければならないという作業手順は，「$at_1(x) \leftarrow at_2(x).$」で記述できる．オブジェクト x がネットリスト抽出やシミュレーションに合格しなければ，x を部品とするオブジェクト yはネットリスト抽出やシミュレーションに合格できないというオブジェクト間の一貫性は，次のようになる．

$$at_1(x) \leftarrow at_1(y), comp(x, y).,$$
$$at_2(x) \leftarrow at_2(y), comp(x, y).$$

以上から，一貫性制約 C_1は次のようになる．

$$\left\{ \begin{array}{l} \leftarrow at_1(x), fg_1(x)., \ \leftarrow at_2(x), fg_2(x)., \\ at_1(x) \leftarrow at_2(x)., \\ at_1(x) \leftarrow at_1(y), comp(x, y)., \\ at_2(x) \leftarrow at_2(y), comp(x, y). \end{array} \right\}$$

例 2で示したように，一貫性制約を用いると状態が各オブジェクトの状況を正しく表していることと状態間の一貫性を保証できるので，$I(\mathcal{DB})$ が一貫性制約を満たすことをデータベースは一貫性を満たすという．形式的には，次のように定義できる．

定義 3 一貫性制約が$\boldsymbol{C}$であるデータベース $DB\langle O, R\rangle$ に対して，$G_C(\mathcal{DB}) =$

$$\left\{ c\theta \;\middle|\; \begin{array}{l} c \in \boldsymbol{C}, \exists\theta = \{x/o_i, y/o_j\}(o_i, o_j \in O, \\ c \text{ 中の } r_k(x, y) \text{ に対して}, r_k(x, y)\theta \in R) \end{array} \right\}$$

中のすべての基礎節が $I(\mathcal{DB})$ において真であれば，$I(\mathcal{DB})$ は一貫性制約を満たす．ここで，代入θとは，変数 x_iに項 t_iを割り当てる有限集合 $\{x_1/t_1, \cdots, x_m/t_m\}$ である．c や $r_k(x, y)$ に代入θを施した結果である$c\theta$や$r_k(x, y)\theta$は，cや$r_k(x, y)$に変数x_iが出現するとき，そのx_iを項t_i $(i = 1, \cdots, m)$に置き換えて得られる表現である． □

$G_C(\mathcal{DB})$ はその論理値を問題にする．$r_k(x, y)\theta = r(o_i, o_j)$ の論理値は $r(o_i, o_j)$ は Rに含まれることから真であるとし，$c\theta$の評価に影響しないので，以下では $r(o_i, o_j)$ を省略する．

例 3 例 1の DB_1と例 2の一貫性制約 C_1において，$G_{C_1}(\mathcal{DB}_1)$ は次のようになる．

$$\left\{ \begin{array}{l} \leftarrow at_1(o_1), fg_1(o_1)., \ \leftarrow at_1(o_2), fg_1(o_2)., \\ \leftarrow at_1(o_3), fg_1(o_3)., \ \leftarrow at_2(o_1), fg_2(o_1)., \\ \leftarrow at_2(o_2), fg_2(o_2)., \ \leftarrow at_2(o_3), fg_2(o_3)., \\ at_1(o_1) \leftarrow at_2(o_1)., \ at_1(o_2) \leftarrow at_2(o_2)., \\ at_1(o_3) \leftarrow at_2(o_3)., \ at_1(o_2) \leftarrow at_1(o_1)., \\ at_1(o_3) \leftarrow at_1(o_1)., \ at_2(o_2) \leftarrow at_2(o_1)., \\ at_2(o_3) \leftarrow at_2(o_1). \end{array} \right\}$$ □

3. 一貫性情報に基づく正当なスケジュール

個々の利用者の$\mathcal{DB}$に対する作業を全順序 $<$ が定義された検索操作 $R(X)$ と変更操作 $W(Y)$ の集合である処理単位 Tで表す．

$$T \subseteq \{R(X), W(Y) \mid X, Y \subseteq \mathcal{DB}\}$$

である．ただし，1つの処理単位中の任意の2つの操作 $R(X), R(Y)$ または $W(X), W(Y)$ に対して，$X \cap Y = \phi$であるとする．$X \cap Y \neq \phi$である $R(X)$,$W(Y)$ に対して，$R(X) < W(Y)$ とする．

複数の処理単位 $T_1, \cdots, T_n$（処理単位集合$\boldsymbol{T}$=$\{T_1, \cdots, T_n\}$）の並行実行をスケジュール $H = \bigcup_{i=1}^{n} T_i$ といい，その実行順序 $<_H$ は $<_H \supseteq \bigcup_{i=1}^{n} <_i$である．異なる処理単位 T_i, T_jに属する $R_i(Y)$ と $W_j(X)$, $W_i(X)$ と $W_j(Y)$, $W_i(X)$ と $R_j(Y)$ $(X \cap Y \neq \phi)$ は競合するという．競合する操作の順序が同じであるスケジュールは等価であるという．Hと等価な直列スケジュール H'が存在すれば，Hは直列可能であると

いう.

並行処理制御の目的は，処理単位の次の4つの性質を保証することである.

- 原子性 (atomicity)：処理単位の操作結果はすべてデータベースに反映されるか，すべて取り消されるかのどちらかでなければならない.
- 整合性 (consistency)：一貫性を満たすデータベースに対して単独実行された処理単位の実行後のデータベースは，一貫したものでなければならない．複数の処理単位を並行実行した場合でも，実行後のデータベースは一貫したものでなければならない.
- 隔離性 (isolation)：処理単位は同時に実行されている他の処理単位の影響を受けず，検索されたデータは一貫したデータベースの部分集合である.
- 耐久性 (durability)：一旦終了した処理単位の操作結果は，その後の障害などで消滅してはならない.

データベースの一貫性が判断できるような一貫性制約$\boldsymbol{C}$を記述できるならば，正当性と隔離性は$\boldsymbol{C}$を用いて形式的に定義できる[4].

定義 4 [4] 一貫性制約が$\boldsymbol{C}$であるデータベースにおいて，処理単位の整合性と隔離性は結果一貫性と検索一貫性として形式化できる.

- 結果一貫性：$\boldsymbol{C}$を満たすデータベースに対して，Tを単独で実行した結果のデータベースは$\boldsymbol{C}$を満たす.
- 検索一貫性：Tの検索されたデータ項目は，$\boldsymbol{C}$を満たすあるデータベースの部分集合である. □

すなわち，処理単位の結果一貫性と検索一貫性を判定するためには，処理単位は単なる操作系列という構造的な定義では不十分で，検索された値と変更した値まで必要とする．構造的に同じである2つの処理単位でも，操作結果によっては結論が異なってくる.

定義 5 処理単位Tに対して，

- $D_R(T) = \bigcup_{R(X)\in T} X$ をTの検索項目，$I_R(T) = \{\langle x, I_R(x)\rangle \mid x \in D_R(T)\}$を$T$の検索値という．$I_R(x)$は$T$が検索した$x$の値である.
- $D_W(T) = \bigcup_{W(X)\in T} X$ をTの変更項目，$I_W(T) = \{\langle x, I_W(x)\rangle \mid x \in D_W(T)\}$を$T$の変更値という．$I_W(x)$は$T$が変更した$x$の値である.
- $D(T) = D_R(T) \cup D_W(T)$ をTの結果項目，$I(T) = (I_R(T) - I_W(T)) \cup I_W(T)$を$T$の結果値という．$\langle x, I(x)\rangle$ $(\in I(T))$で，$I(x)$は，$x \in D_R(T) - D_W(T)$のときTが検索したxの値$I_R(x)$，$x \in D_W(T)$のときTが変更したxの値$I_W(x)$である.

また，利用者は直接$fg_k(o)$を変更しないが，処理単位の$e(o)$または$at_k(o)$に対する変更操作によって$fg_k(o)$の値が自動的に変更されるので，その値は結果値に含まれるものとする. □

すなわち，あるデータ項目が処理単位Tによって検索と変更の両方がなされている場合には，変更値を結果値とする.

	T_1	データ項目の値
t_1	$R_1(\{e(o_2)\})$	
t_2	$R_1(\{at_1(o_2)\})$	$\langle at_1(o_2), t\rangle$
t_3	$W_1(\{at_2(o_2)\})$	$\langle at_2(o_2), t\rangle$
t_4	$R_1(\{at_1(o_1)\})$	$\langle at_1(o_1), t\rangle$
t_5	$R_1(\{at_2(o_3)\})$	$\langle at_2(o_3), t\rangle$
t_6	$R_1(\{e(o_1)\})$	
t_7	$W_1(\{at_2(o_1)\})$	$\langle at_2(o_1), t\rangle$

図1 操作結果も明示される処理単位T_1
Fig. 1 Transaction T_1 with its operated result

例 4 例1のデータベースにおける図1のような順でo_2,o_1に対してシミュレーションを行う処理単位T_1について考察する.

T_1の検索項目$D_R(T_1)$，変更項目$D_W(T_1)$や検索値$I_R(T_1)$等は次のようになる.

$$D_R(T_1) = \{e(o_2), at_1(o_2), at_1(o_1), at_2(o_3), e(o_1)\},$$
$$D_W(T_1) = \{at_2(o_2), at_2(o_1)\},$$
$$I_R(T_1) = \{\langle at_1(o_2), t\rangle, \langle at_1(o_1)), t\rangle, \langle at_2(o_3)), t\rangle\},$$
$$I_W(T_1) = \{\langle at_2(o_2), t\rangle, \langle at_2(o_1), t\rangle\},$$
$$I(T_1) = \left\{ \begin{array}{l} \langle at_1(o_2), t\rangle, \langle at_1(o_1)), t\rangle, \langle at_2(o_3)), t\rangle, \\ \langle at_2(o_2), t\rangle, \langle at_2(o_1), t\rangle, \langle fg_2(o_1), f\rangle, \\ \langle fg_2(o_2), f\rangle \end{array} \right\}.$$
□

従来では隔離性と並行実行の整合性は，スケジュールの直列可能性によって保証されていた．しかし，並行実行の整合性のみを保証するためには，直列可能性を必要としない.

定理 1 [4] 処理単位集合$\boldsymbol{T}$のスケジュールHに対して，各$T \in \boldsymbol{T}$が結果一貫性を満たし，判定グラフ$DG(H)$に閉路が存在しなければ，Hの実行後のデータベースは一貫性を満たす．$DG(H)$の節点は処理単位T_iの操作であり，枝は次のものである.

(1) 処理単位間R枝：$W_i(X) <_H R_j(Y)(X \cap Y \neq \phi, i \neq j)$ のとき，枝 $(W_i(X), R_j(Y))$.

(2) 処理単位間W枝：$R_i(X) <_H W_j(Y)$ または $W_i(X) <_H W_j(Y)$ $(X \cap Y \neq \phi,\ i \neq j)$ のとき，枝 $(U, W_j(Y))$ (UはT_iのすべての操作).

(3) 処理単位内枝：$U <_i V$ $(U, V \in T_i)$ のとき，枝 (U, V). □

$DG(H)$ に閉路が存在しないスケジュールを論理可能という．論理可能スケジュール H に対して，H と等価で処理単位 $T \in \boldsymbol{T}$ が終了するまで操作されたデータ項目が $\boldsymbol{T}$ の他の処理単位によって変更されないような H' が存在する[5]．検索一貫性と結果一貫性を満たす処理単位からなる処理単位集合 $\boldsymbol{T}$ のスケジュール H は，論理可能であれば，独立化可能であるという．独立化可能スケジュールは処理単位の整合性と隔離性および並行実行の整合性を保証できるので，正当なスケジュールである[4]．また，論理可能スケジュールは，直列可能スケジュールと比較して検索操作に対する制限が緩和されている[5]．

4. 結果一貫性の分割判定

一貫性情報を利用できるならば，一貫性制御にどのような影響を与えるかについて検討する．

利用者が操作するデータはデータベースの一部なので，データベースの部分集合が $\boldsymbol{C}$ を満すかどうかを考えるため，$\boldsymbol{C}$ のデータ項目の部分集合による基礎展開を定義する．

定義 6 データベース $DB\langle O, R\rangle$ に対して，一貫性制約 $\boldsymbol{C}$ のデータ項目集合 D $(D \subseteq \mathcal{DB})$ による基礎展開 $G_C(D)$ は，$G_C(\mathcal{DB})$ の次のような部分集合である．

$$G_C(D) = \{c\theta \mid c\theta \in G_C(\mathcal{DB}), \Gamma(c\theta) \cap D \neq \phi\}$$

ここで，$\Gamma(c\theta)$ は基礎節 $c\theta$ に現れる状態の集合である． □

例 5 例1の DB_1 と例2の一貫性制約 C_1 において，$G_{C_1}(\{at_2(o_1)\}) =$

$$\left\{ \begin{array}{ll} \leftarrow at_2(o_1), fg_2(o_1)., & at_1(o_1) \leftarrow at_2(o_1)., \\ at_2(o_2) \leftarrow at_2(o_1)., & at_2(o_3) \leftarrow at_2(o_1). \end{array} \right\}$$

であり，$G_{C_1}(\{at_2(o_2)\}) =$

$$\left\{ \begin{array}{ll} \leftarrow at_2(o_2), fg_2(o_2)., & at_1(o_2) \leftarrow at_2(o_2)., \\ at_2(o_2) \leftarrow at_2(o_1). & \end{array} \right\}$$

である． □

結果一貫性の判定問題については，一貫性制約の基礎展開の定義より次のことが分かる．

定理 2 処理単位集合 $\boldsymbol{T}$ のスケジュール H において，処理単位 T $(\in \boldsymbol{T})$ の変更項目 $D_W(T)$ による一貫性制約 $\boldsymbol{C}$ の基礎展開 $G_C(D_W(T))$ 中のすべての基礎節が，T の結果値 $I(T)$ において真であれば，T は結果一貫性を満たす． □

証明：基礎展開の定義により，T が単独で実行された場合，それによって真偽が変わる可能性のある $\boldsymbol{C}$ 中の制約を基礎展開した基礎節は，すべて $G_C(D_W(T))$ に含まれている．このため，$G_C(D_W(T))$ が $I(T)$ において真であるならば，$\boldsymbol{C}$ を満たすデータベースで T が単独で実行された場合，結果のデータベースも $\boldsymbol{C}$ を満たす．したがって，T は結果一貫性を満たす． □

例 6 例2の C_1 を一貫性制約とする例1の DB_1 において，例4と例5から $G_{C_1}(D_W(T_1)) =$

$$\left\{ \begin{array}{ll} \leftarrow at_2(o_1), fg_2(o_1)., & \leftarrow at_2(o_2), fg_2(o_2)., \\ at_1(o_1) \leftarrow at_2(o_1)., & at_1(o_2) \leftarrow at_2(o_2)., \\ at_2(o_2) \leftarrow at_2(o_1)., & at_2(o_3) \leftarrow at_2(o_1). \end{array} \right\}$$

となる．$G_{C_1}(D_W(T_1))$ は

$$I(T_1) = \left\{ \begin{array}{l} \langle at_1(o_2), t\rangle, \langle at_1(o_1)), t\rangle, \langle at_2(o_3)), t\rangle, \\ \langle at_2(o_2), t\rangle, \langle at_2(o_1), t\rangle, \\ \langle fg_2(o_1), f\rangle, \langle fg_2(o_2), f\rangle \end{array} \right\}$$

において真なので，T_1 は結果一貫性を満たす． □

定理2より，結果一貫性は基礎展開 $G_C(D_W(T))$ 中の基礎節の真偽を調べることによって判定できる．基礎節 $p_1, \cdots, p_s \leftarrow q_1, \cdots, q_t.$ は，$I(q_i)$ が偽である q_i $(1 \leq i \leq t)$ または $I(p_j)$ が真である p_j $(1 \leq j \leq s)$ が存在するとき真なので，結果一貫性の判定時間は検査される基礎節の数の多項式時間である．一貫性制約 $\boldsymbol{C}$ に含まれる節の数を l，1つのオブジェクトに関するオブジェクトの数を r とすると，結果一貫性の判定のために検査される基礎節の数は $\mid D_W(T) \mid \times l \times r$ となり，結果一貫性の判定は $O(\mid D_W(T) \mid \times l \times r)$ という多項式時間となる．通常 l は定数であり，1つのオブジェクトと関連する履歴関連や部品関連などのオブジェクト間の関連数 r も定数と見なせる．このため，結果一貫性の判定は処理単位の変更されるデータ項目の数 $\mid D_W(T) \mid$ に対して $O(\mid D_W(T) \mid)$ 時間という実用的な時間となる．

処理単位内のある操作 $u \in T$ に対して，u と u より

前のTの操作からなる$T_u = \{u\} \cup \{v \mid v < u, v \in T\}$は，$T$の部分処理単位となる．処理単位の結果一貫性の判定が実用的な時間で判定できるならば，部分処理単位T_uの結果一貫性も同じ方法で判定できる．

例 7 例4において，部分処理単位$T_{W_1(at_2(o_2))}$の結果一貫性について考察する．

$G_{C_1}(D_W(T_{W_1(at_2(o_2))}))=G_{C_1}(\{at_2(o_2))\}) =$

$$\left\{ \begin{array}{l} \leftarrow at_2(o_2), fg_2(o_2)., at_1(o_2) \leftarrow at_2(o_2)., \\ at_2(o_2) \leftarrow at_2(o_1). \end{array} \right\}$$

となる．$G_{C_1}(D_W(T_{W_1(at_2(o_2))}))$は$I(T_{W_1(at_2(o_2))}) =$

$\{(at_1(o_2), t), \langle at_2(o_2), t\rangle, \langle fg_2(o_2), f\rangle\}$

において真なので，$T_{W_1(at_2(o_2))}$は結果一貫性を満たす．したがって，T_1の分割数が2である． □

定理1より，論理可能スケジュールにおいて，結果一貫性を満たす部分処理単位の終了時点のデータベースは一貫したものである．したがって，処理単位に検索された後で変更され，部分処理単位の結果一貫性を満たしたデータ項目に対しては，その変更値を利用すればよい．すなわち，検索したデータ項目の値が一貫している代わりに，それから部分処理単位の結果一貫性を満たした項目を除いたものが一貫していればよい．

定義 7 処理単位Tの各部分処理単位T_uに対して，$I_R(T_u) - I_W(T_u)$がある一貫したデータベースの部分集合ならば，Tは分割検索一貫性を満たすという． □

したがって，分割判定を行っている処理単位集合のスケジュールに対して，正当なスケジュールの範囲を次のように拡大できる．

定義 8 分割検索一貫性と結果一貫性を満たす処理単位からなる処理単位集合$\boldsymbol{T}$の論理可能スケジュールHは，分割判定独立化可能という． □

定理 3 結果一貫性を満たす処理単位集合$\boldsymbol{T}$の論理可能スケジュールHにおいて，各$T_i \in \boldsymbol{T}$の検索するデータ項目が結果一貫性を満たす(部分)処理単位の変更結果であれば，Hは分割判定独立化可能である．

証明：論理可能スケジュールに対して，定理1より，結果一貫性を満たす各(部分)処理単位の終了時点のデータベースは$\boldsymbol{C}$を満たす．各$T_i \in \boldsymbol{T}$の検索するデータ項目が結果一貫性を満たす(部分)処理単位の変更結果なので，各処理単位の検索操作はそれぞれ一貫したデータベースを検索する．したがって，定義8より処理単位Tが分割検索一貫性を満たすかどうかは，各部分処理単位T_uの検索されたデータ項目を変更する他の処理単位の変更操作が存在するかどうかに帰する．

論理可能スケジュールHに対して，処理単位$T \in \boldsymbol{T}$が終了するまで操作されたデータ項目が$\boldsymbol{T}$の他の処理単位によって変更されないような等価なH'が存在する．したがって，H'においてTは分割検索一貫性を満たす．HはH'と等価なので，HにおいてもTは分割検索一貫性を満たすので，Hは分割判定独立化可能である． □

	T_2	T_3	操作結果
t_1	$R_2(\{e(o_2)\})$		
t_2		$R_3(\{at_1(o_3)\})$	$\langle at_1(o_3), t\rangle$
t_3	$W_2(\{at_1(o_2)\})$		$\langle at_1(o_2), t\rangle$
t_4		$R_3(\{at_1(o_2)\})$	$\langle at_1(o_2), t\rangle$
t_5		$R_3(\{e(o_1)\})$	
t_6	$W_2(\{at_2(o_2)\})$		$\langle at_2(o_2), t\rangle$
t_7	$R_2(\{at_1(o_1)\})$		$\langle at_1(o_1), t\rangle$
t_8		$W_3(\{at_1(o_1)\})$	$\langle at_2(o_1), t\rangle$
t_9	$R_2(\{at_2(o_3)\})$		$\langle at_2(o_3), t\rangle$
t_{10}	$R_2(\{e(o_1)\})$		
t_{11}	$W_2(\{at_2(o_1)\})$		$\langle at_2(o_1), t\rangle$

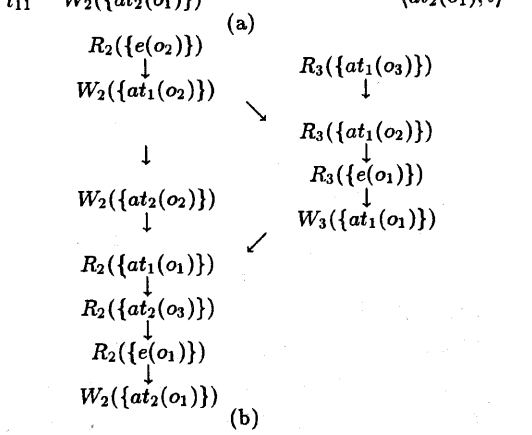

図2 非直列可能な分割判定独立化可能スケジュールH_1
Fig. 2 non-serializable but Equivalent-independent schedule H_1

例 8 DB_1において，図2 (a)のような互いに操作結果を参照しながら作業していくスケジュールH_1について考察する．判定グラフ$DG(H_1)$(図2 (b))の枝はすべて時間順に従っており，閉路は存在しないので，H_1は論理可能である．また，T_2の変更操作ごとに分割した部分処理単位は結果一貫性を満たしており，T_3も結果一貫性を満たしている．したがって，すべての検索操作は結果一貫性を満たす変更結果を検索することになり，H_1は分割判定独立化可能である．

しかし，$W_2(\{at_1(o_3)\}) <_{H_1} R_3(\{at_1(o_3)\})$ と $W_3(\{at_1(o_1)\}) <_{H_1} R_2(\{at_1(o_1)\})$ より，H_1は直列スケジュールT_2T_3やT_3T_2とは等価ではなく，直列可能ではない． □

5. 分割数と処理単位構造の関連

結果一貫性を満たす部分処理単位数が多ければ，早い時期に他の処理単位が変更結果を検索できるようになる．逆にすべての処理単位に対して分割数が1であれば，実行できるスケジュールは直列可能スケジュールのみである．一方，実用化に向けては無駄な分割判定によるコストも考慮しなければならない．

	T_1	T_4
t_1	$R_1(\{e(o_2)\})$	$R_4(\{e(o_1)\})$
t_1	$R_1(\{at_1(o_2)\})$	$R_4(\{at_1(o_2)\})$
t_2	$W_1(\{at_2(o_2)\})$	$W_4(\{at_2(o_1)\})$
t_3	$R_1(\{at_1(o_1)\})$	$R_4(\{at_1(o_1)\})$
t_4	$R_1(\{at_2(o_3)\})$	$R_4(\{at_2(o_3)\})$
t_1	$R_1(\{e(o_1)\})$	$R_4(\{e(o_2)\})$
t_5	$W_1(\{at_2(o_1)\})$	$W_4(\{at_2(o_2)\})$

図3 実行順序のみが異なる T_1 と T_4

Fig. 3 Transactions where $T_1 = T_4$ but $<_1 \neq <_4$

例9 DB_1において，図3で示した操作集合と操作結果は同じであるが実行順序が異なる処理単位 T_1，T_4 について考察する．

例7で示したように T_1の分割数は2である．しかし，$T_{W_4(at_2(o_1))}$において，o_2がシミュレーションに合格しなければ，o_2を部品とする o_1はシミュレーションに合格できないという制約 $at_2(o_2) \leftarrow at_2(o_1). \in G_{C_1}(D_W(T_{W_4(at_2(o_1))}))$ は $I(T_{W_4(at_2(o_2))})$ において真偽が確認できないので，$T_{W_4(at_2(o_1))}$は結果一貫性を満たさず，T_4 は分割判定できない． □

例9が示すように，分割判定の状況は処理単位の実行順序に依存し，最も細分化できる場合は変更操作ごとに分割した部分処理単位が結果一貫性を満たす．

処理単位の実行順序の性質を検討するために，基礎展開に関する次の記号を導入する．

定義9 基礎展開 $G_C(\mathcal{DB})$ による状態間の二項関連は次のものである．

順序関連：IS=

$$\left\{(p_i,q_j)\,\middle|\, \begin{array}{l} p_1,\cdots,p_s \leftarrow q_1,\cdots,q_t. \in G_C(\mathcal{DB}) \\ 1 \le i \le s, 1 \le j \le t \end{array}\right\}$$

並列関連：EX=

$$\left\{(q_i,q_j)\,\middle|\, \begin{array}{l} p_1,\cdots,p_s \leftarrow q_1,\cdots,q_t. \in G_C(\mathcal{DB}) \\ 1 \le i,j \le t \end{array}\right\}$$

交替関連：AL=

$$\left\{(p_i,p_j)\,\middle|\, \begin{array}{l} p_1,\cdots,p_s \leftarrow q_1,\cdots,q_t. \in G_C(\mathcal{DB}) \\ 1 \le i,j \le s \end{array}\right\}$$ □

順序関連はオブジェクトの状態の値が偽から真になる順序を定めていると考えることができる．状態を節点，順序関連 (p_i,q_j) を枝とする有向グラフをIS関連グラフとすると，IS関連グラフは$\mathcal{DB}$における作業手順を表していることになる．このため，IS関連グラフには閉路は存在しないものとする．

例10 DB_1のデータ項目集合$\mathcal{DB}_1$におけるIS関連グラフは図4となる． □

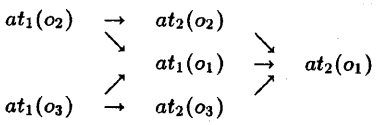

図4 IS関連グラフ

Fig. 4 IS relationship graph

定義10 処理単位Tは，次の条件を満たすとき一貫性制約Cに対して整構造であるという．

(1) $(x,y) \in EX \cap AL$ ならば，x と yに対する変更は1つの変更操作 $W(\{x,y,\cdots\})$ で行われる．

(2) Tによって値が変更される状態のデータ項目は，IS関連グラフのある位相的整列 (topological sorting) の昇順に従って真に，降順に従って偽に変更される．

(3) 状態 x を真にする変更操作 $W(X)(x \in X)$ と状態 yを偽にする変更操作 $W(Y)(y \in Y)$ に対して，$(x,y) \in EX - AL$ ならば $W(Y) < W(X)$ であり，$(x,y) \in AL - EX$ ならば $W(X) < W(Y)$ である．

(4) 実体 $e(o)$ に対する変更操作 $W(X)$ $(e(o) \in X)$ に対して，$at_k(o)$ を変更操作 $W(Y)(at_k(o) \in Y)$ $(1 \le k \le m)$ が値を真にするのであれば $W(X) < W(Y)$ である．

(5) データ項目 x,yが基礎展開 $G_C(D_W(T))$ 中の同じ節の項であるならば，$x \in X$である検索操作 $R(X)$ と $y \in Y$である変更操作 $W(Y)$ に対し，$R(X) < W(Y)$ である． □

例 11 図 4から $at_2(o_2)$ と $at_2(o_1)$ に関する位相的整列の昇順には $at_2(o_2)$ の方が先であることが分かる．分割数 2 の T_1の実行順序はそれに従っており，分割数 1 の T_2の実行順序はその逆順である． □

補題 1 結果一貫性を満たす整構造である処理単位 T において，任意の h 番目の変更操作α_hで分割した部分処理単位 T_{α_h}は結果一貫性を満たす．

証明：整構造の定義より，$W(X)$ の分割判定に必要なデータ項目に対する検索操作は，$W(X)$ が実行されるまでに行われる．閉路がないグラフでは節点の位相的整列が存在するので，条件 (2) と (5) で $G_C(D_W(T))$ 中の基礎節の頭部 (head) と体部 (body) に表れる状態述語に対する操作間の実行順序のために T_{α_h}が結果一貫性を満たさなくなることはない．

任意の $W(X)(x \in X)$，$W(Y)(y \in Y)$ に対して，$(x, y) \notin EX \cap AL$ であることより，条件 (3) と (5) でそれぞれ $G_C(D_W(T))$ 中の基礎節の頭部 (head) または体部 (body) に表れる状態述語に対する操作間の実行順序のために T_{α_h}が結果一貫性を満たさなくなることはない．一方，条件 (4) の状態に対する変更操作と実体に対する変更操作の実行順序で，T_{α_h}は実体と状態間の結果一貫性を満たすことを保証している．よって T_{α_h}は結果一貫性を満たす． □

変更操作の要素が 1 つである任意の処理単位に対して，整構造の条件を満たさない変更操作のみを 1 つにまとめた後，定義 10に従って操作間の実行順序を決めれば，操作内容を変えずに整構造に変換することができる．

系 1 結果一貫性を満たす整構造処理単位集合$\boldsymbol{T}$のスケジュール Hは，論理可能ならば，分割判定独立化可能である．

証明：補題 1より，任意の変更操作によって分割される部分処理単位は結果一貫性を満たす．したがって，各 $T_i \in \boldsymbol{T}$の検索するデータ項目は結果一貫性を満たす (部分) 処理単位の変更結果である．定理 3より，そのようなスケジュールは分割判定独立化可能である． □

直列可能性の主な問題は検索操作に対する制限にあること，論理可能性は検索操作に対する制限を緩和していること，スケジュールの論理可能性は多項式時間で判定できることにより，整構造処理単位からなる分割判定独立化可能クラスは並行処理制御の実用的な正当性クラスとなりえる．

回復制御の点から，分割判定独立化可能スケジュールの保証方法は，処理単位の一部分を後退復帰する部分後退復帰を可能とする．これは，この十分条件の下で部分後退復帰しても，処理単位は一貫したデータベースから一貫したデータベースへの遷移であるという性質を保証できるためである．

6. おわりに

本論文では，一貫性情報を利用できれば，処理単位の ACID 特性が形式的に定義でき，処理単位と部分処理単位の結果一貫性問題が多項式時間で判定できることを示した．次に，部分処理単位の結果一貫性を満たした変更結果を検索することで，検索一貫性を保証できる方法を与え，分割判定独立化可能性という実用化に向けたクラスを提案した．さらに，分割判定の性質を分析するために，処理単位の構造を検討し，整構造というそのための構造を提案した．結果一貫性を満たす整構造処理単位からなるスケジュールは，論理可能であれば，分割判定独立化可能である．論理可能性は直列可能性と比較して検索操作に対する制限が緩和されているので，分割判定独立化可能性は，協調型データベースにおける並行実行での問題を解決していることになる．

参考文献

1) Barghouti, N. S. and Kaiser, G. E.: Concurrency Control in Advanced Database Applications, *ACM Comput. Surv.*, Vol. 23, No. 3, pp. 269-317 (1991).
2) Bernstein, P. A., Hadzilacos, V., and Goodman, N.: *Concurrency Control and Recovery in Database Systems*, Addison-Wesley (1987).
3) Elmagarmid, A. K.: *Database Transaction Models for Advanced Applications*, Morgan Kaufmann, 1992.
4) 徐海燕，古川哲也，史一華：一貫性情報を用いたデータベースの並行処理制御，情報処理論文誌，Vol. 35, No. 12, pp. 2752-2761 (1994).
5) 徐海燕，古川哲也，史一華：並行処理制御方式による独立化可能クラスと直列可能クラスの比較，情報処理論文誌，Vol. 37, No. 8, pp. 1600-1609 (1996).