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

徐 海燕[†] 古川哲也[††] 史 一華[†††]

高水準データベース（DB）における並行処理制御への新しい要求に対応するため，独立化可能というDBの持つ一貫性情報を利用した正当なクラスが提案され，それは従来の直列可能クラスより範囲を拡大したクラスであることが示されている．スケジュールの独立化可能性の判定は，論理性に関する問題と一貫性に関する問題という2つの問題に分けられる．本論文では前者の問題に焦点を当て，既存の2相施錠方式，時刻印方式，グラフ判定方式の変更版を導入し，変更版と元の方式の相違により，独立化可能クラスの範囲を拡大できた部分の性質を示す．

Comparing Equivalent-independent and Serializable Classes by Concurrency Control Protocols

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

Based on the integrity constraints of advanced databases, a new criterion of schedules called equivalent-independence has been proposed, which is more generous than the serializability criterion. In this paper, we discuss the concurrency control protocols for the new criterion, which cover three typical kinds: two phase locking, timestamp ordering, and graph testing. By comparing the concurrency control protocols for these two criteria, we show the features of the new criterion.

1. はじめに

データベース（DB）システムの全体の利用効率を向上させるためには，複数個の処理単位を並行して扱う必要がある．しかし，各々の処理単位に用いられるデータに重なりがあれば，共有データによる処理単位間の干渉によって種々の問題が生じる．それを防止および解決するために，並行処理制御に関する研究が行われている．従来の事務DBにおいては，直列可能である（実行結果がある順序での直列実行と等価となる）スケジュールのクラスが並行処理制御の正当なクラスとされており，それに対する種々の並行処理制御方式が提案されている[4),9),10)]．

しかし，協同作業や長時間の処理を支援する高水準DBにおいては，直列可能なスケジュールのみでは満足できる並行実行を提供できない[3),6)]．このため，様々な角度から研究が行われてきており[2),3),7)]，DBの持つ一貫性制約を利用することにより，独立化可能という直列可能より範囲を拡大した正当なクラスが提案されている[11)]．独立化可能なスケジュールのクラスは，各処理単位が次の論理性，検索一貫性，結果一貫性という3つの性質を満たす等価なスケジュールが存在するものからなる．

- 論理性：各処理単位によって操作されたデータ項目は，その処理単位が終了するまで，他の処理単位によって変更されていない．
- 検索一貫性：各処理単位によって検索されるのは，一貫性制約を満たしたあるDBの部分集合である．
- 結果一貫性：各処理単位によって変更されたDBは一貫性制約を満たす．

このため，スケジュール H の独立化可能性を判定する問題は，次の2つの問題に分けられる．

- 論理性に関する問題：H と等価で各処理単位は論理性を満たすようなスケジュール H' が存在するかどうかの問題．
- 一貫性に関する問題：H 中の各処理単位の検索結果による一貫性制約の充足可能性問題と変更結果に対する一貫性検証問題．

† 福岡工業大学工学部電子工学科
Department of Electronics, Fukuoka Institute of Technology
†† 九州大学経済学部経済工学科
Department Economic Engineering, Kyushu University
††† 福岡工業短期大学
Fukuoka Junior College of Technology

論理性に関する問題はDBや操作の内容には関わらず，与えられたスケジュールによって一意に決められるが，一貫性に関する問題はDBの持つ一貫性制約と各処理単位の操作結果に依存する．極端にいえば，DBの持つ一貫性制約が空ならば，論理性に関する問題のみで独立化可能性が判定できる．一般に，与えられた一貫性制約に対して，変更結果の一貫性検証に必要なデータ項目を検索した後に変更を行えば，結果一貫性を満たすかどうかは各処理単位の責任になる．したがって，各々の処理単位の実行はDBの一貫性を保つという直列可能性基準における仮定と同様に，各々の処理単位の責任で結果一貫性を満たすという仮定を設けることも考えられる．また，検索一貫性に対してはマルチバージョン制御法[4]を用いて満たす方法が考えられる．すなわち，論理性に関する問題のみでスケジュールの独立化可能性を判定できると考えられる．このため，そのようなスケジュールの性質を検討することは独立化可能なスケジュールの性質を明らかにするうえできわめて重要となる．論文11)では，統一的な判定基準の下でのスケジュールの正当性基準の検討が主な内容であり，独立化可能クラスは直列可能クラスより範囲を拡大したものであることを示したところにとどまっている．本論文では，論理性に関する問題について，この拡大された部分によって直列可能性のための並行処理制御方式における問題点がどのように解決されているのかを具体的に示す．

直列可能性のための並行処理制御方式については，数多くの方式が提案されている[3),4)]．それらは主に施錠を用いる方式，後退復帰を用いる方式という2種類に分けられる[1),4)]．施錠方式では，各検索操作または変更操作を行う前に操作される各要素に対して共有施錠または専有施錠を行い，かつすでに専有施錠されているデータ項目には他の処理単位は再び施錠できず，すでに共有施錠されているデータ項目には他の処理単位は再び専有施錠できない．長時間の処理に対して，施錠が数日に渡る可能性があるので，施錠方式には耐えられない遅延を招く問題が存在する．一方，後退復帰方式である時刻印方式では，各処理単位の到着順に時刻印を割り当て，所属する処理単位の時刻印が増える順での競合操作の実行のみを許可することで直列可能性を保証する．すなわち，それを満たさない検索操作または変更操作が実行されれば，所属する処理単位は後退復帰される．グラフ判定方式も後退復帰方式であり，オンライン判定グラフを管理することによってスケジュールが直列可能性を満たしているかどうかを直接判定する．検索操作または変更操作の実行によって判定グラフに閉路が生じるならば，直ちに所属する処理単位は後退復帰される．したがって，長時間の処理に対して，後退復帰方式でも長時間操作結果の紛失問題が起こる．本論文では，2相施錠方式，時刻印方式，グラフ判定方式という最もよく知られている方式に対して，論理性に関する問題のための変更版を与える．各変更版によって上記の問題がどのように解決されているかを示すとともに，直列可能性のための方式との比較により，並行処理制御方式の視点から独立化可能クラスの範囲を拡大できた部分の性質をまとめる．

本論文は，次のように構成される．2章では，直列可能性とD-直列可能性の定義や並行処理制御に関する基本的事項を与える．3章では，独立化可能性の3つの性質中の論理性に焦点を当て，そのための判定グラフなどを説明する．4章と5章では，既存の2相施錠方式と時刻印方式の変更版を検討し，変更版が元の方式より拡張できた部分の特徴を分析する．6章では，判定グラフによる比較を行う．7章は，全体のまとめである．

2. 基本的事項

本章では，データ項目集合 V を持つDBにおける並行処理制御に関する基本的事項を定める．

定義1 処理単位 $T_i = \{R_i(X), W_i(X) \mid X \subseteq V\} \cup \{f_i\}$ は，次のような半順序関連 $<_i$ を満たす実行順序を持つ操作集合である．ただし，$R_i(X)/W_i(X)$ で T_i の X に対する検索/変更操作，f_i で T_i の終了操作を表す．

- f_i 以外の $p \in T_i$ に対して，$p <_i f_i$ である．
- $R_i(X), R_i(Y) \in T_i$，または $W_i(X), W_i(Y) \in T_i$ なら，$X \cap Y = \phi$ である．
- $R_i(X), W_i(Y) \in T_i (X \cap Y \neq \phi)$ なら，$R_i(X) <_i W_i(Y)$ である． □

定義2 $\boldsymbol{T} = \{T_1, T_2, \cdots, T_n\}$ を処理単位の集合とすると，$\boldsymbol{T}$ 上のスケジュール H は，次のような半順序関連 $<_H$ を満たす実行順序を持つ操作集合である．

(1) $H = \cup_{i=1}^{n} T_i$.

(2) $<_H \supseteq \cup_{i=1}^{n} <_i$.

(3) $(X \cap Y \neq \phi) \wedge (i \neq j)$ のような競合操作 $A_i(X)$ と $W_j(Y)$ に対して，$A_i(X) <_H W_j(Y)$ または $W_j(Y) <_H A_i(X)$ である．ただし，$A \in \{R, W\}$ である． □

スケジュール H を分類するために，次のような概念が用いられている．

定義3 処理単位集合 $\boldsymbol{T} = \{T_1, T_2, \cdots, T_n\}$ 上の2つのスケジュール H と H' は次の2つの条件を満た

すとき等価であるといい，$H \equiv H'$ で表す[5),8)]．

(1) 任意の $T_i \in \boldsymbol{T}$に対して，H において T_i によって検索されるデータ項目の値と，H' において T_i によって検索されるデータ項目の値は同一である．

(2) 任意の $T_0 = W_0(V)$ に対して，$W_0(V)HR_f(V)$ において $T_f = R_f(V)$ によって検索されるデータ項目の値と，$W_0(V)H'R_f(V)$ において $T_f = R_f(V)$ によって検索されるデータ項目の値は同一である．□

H は，ある直列スケジュール H' と等価であれば，直列可能であるという．しかし，直列可能性を判定する問題はNP完全である[4),5),8)]ため，D-等価の下での多項式時間で判定可能なD-直列可能性（競合保存直列可能性（conflict serializability）ともいう[4)]）部分クラスが利用されている[8)]．

定義4 $\boldsymbol{T}= \{T_1, T_2, \cdots, T_n\}$ 上のスケジュール H が次のような隣合う操作の入れ替え（D-等価交換）で H' に変換できるとき，H と H' はD-等価であるといい，$H \sim H'$ で表す．

(1) $R_i(X)R_j(Y)$

(2) $R_i(X)W_j(Y)(i \neq j, X \cap Y = \phi)$

(3) $W_i(X)W_j(Y)$, $W_i(X)R_j(Y)(X \cap Y = \phi)$ □

$\sim^*$ を $\sim$ の反射的推移閉包とすると，与えられた $\boldsymbol{T}$上の H が，ある直列スケジュール H' に対して $H \sim^* H'$ であれば，H はD-直列可能であるという．

定義5 スケジュール H のD-直列可能性の判定グラフ $D(H)$ は，各 $T_i \in \boldsymbol{T}$に対応して1つの節点 T_i を持ち，T_i から T_j $(i \neq j)$ への枝は，次の条件が満たされるときに作られる．

(1) $W_i(X) <_H R_j(Y)(X \cap Y \neq \phi)$

(2) $A_i(X) <_H W_j(Y)(A \in \{R, W\}, X \cap Y \neq \phi)$ □

定理1[8)] スケジュール H のD-直列可能性の判定グラフ $D(H)$ に閉路が存在しなければ，$H' \sim^* H$ である直列スケジュール H' が存在する． □

D-直列可能であるスケジュールは直列可能であるのは自明であるが，直列可能でD-直列可能でないものが存在することが知られている．D-直列可能性のための並行処理制御方式として，判定グラフを検査することによってD-直列可能性を保証するグラフ判定（GT）方式以外に，施錠方式と後退復帰方式の中で最もよく知られているのは，次の2相施錠方式と時刻印方式である．

定義6[4)] 2相施錠（2PL）方式とは，各 $R_i(X) \in H$ または $W_i(X) \in H$ を行う前に X 中の各要素に対して共有施錠または専有施錠を行い，かつ各処理単位 $T_i \in \boldsymbol{T}$は施錠部分と解錠部分とに分かれる並行処理制御方式である．ただし，すでに専有施錠されているデータ項目には他の処理単位が再び施錠することはなく，すでに共有施錠されているデータ項目には他の処理単位が再び専有施錠することはない．2PL方式に従うスケジュール H を2PLスケジュールという． □

定義7[4)] 時刻印（TO）方式とは，各処理単位 $T_k \in \boldsymbol{T}$の到着順に割り当てられる時刻印 $ts(T_k)$ と各々の T_i と T_j に属する競合操作 p, q $(p \in T_i, q \in T_j, i \neq j)$ に対して，$ts(T_i) < ts(T_j)$ ならば $p <_H q$ であるように操作の実行順序を要求する並行処理制御方式である．TO方式に従うスケジュール H をTOスケジュールという． □

この2つの方式は，D-直列可能性のための並行処理制御方式である．

定理2[4)] 2PLスケジュールおよびTOスケジュールは，D-直列可能である． □

3. 独立化可能性と $L_D(\boldsymbol{T})$ クラス

本章では，独立化可能性という一貫性制約 $\boldsymbol{C}$で表されるDBの一貫性の統一的な判定基準の下でのスケジュールの正当性に関する必要事項を述べるとともに，一貫性制約 $\boldsymbol{C}$に依存しないクラス $L_D(\boldsymbol{T})$ の定義を与える．

直列可能性に基づく並行処理制御では，直列可能であるスケジュールの実行のみを許可する方法でDBの一貫性を保証しているが，設計DBなどの高水準DBにおいて，一貫したDBとは一貫性制約 $\boldsymbol{C}$を満たすDBであるというようにDBの一貫性の統一的な判定基準を定められることが示されている[11)]．一貫性の判定基準を定めることができれば，次のような独立化可能なスケジュール H でDBの一貫性を保証できる．

定義8[11)] 一貫性制約 $\boldsymbol{C}$を持つDBにおいて独立スケジュール $H' \equiv H$ が存在すれば，H は**独立化可能**である．$\boldsymbol{T}$上のスケジュール H が独立であるとは，各 $T_i \in \boldsymbol{T}$が次の3つの性質を満たすことをいう．

(1) 論理性：T_i によって操作されたデータ項目は，T_i が終了するまで，他の $T_j (j \neq i)$ によって変更されていない．すなわち，すべての $A_i(X) <_H W_j(Y)$ $(i \neq j, X \cap Y \neq \phi, A \in \{R, W\})$ に対して，$f_i <_H W_j(Y)$ である．

(2) 検索一貫性：T_i によって検索されるデータ項目は，ある一貫したDBの部分集合である．

(3) 結果一貫性：T_i によって変更されたDBは一貫性制約 $\boldsymbol{C}$を満たす． □

独立化可能なスケジュール H は，各 $T_i \in \boldsymbol{T}$が検索するDBは一貫していること，および T_i は一貫したDBから一貫したDBへの独立遷移であることと，H は一貫したDBから一貫したDBへの遷移であることを保証できるので，利用者に対してもDBシステムに対しても正当なスケジュールである[11]．

一方，スケジュールの独立化可能性を判定する問題は，次のようになる．

定義9 一貫性制約 $\boldsymbol{C}$を持つDBと $\boldsymbol{T}$上のスケジュール H に対して，その独立化可能性を判定する問題は次の2つの問題に分けられる．

- 論理性に関する問題：H と等価で各 $T_i \in \boldsymbol{T}$は論理性を満たすようなスケジュール H' が存在するかどうかの問題．
- 一貫性に関する問題：一貫性制約 $\boldsymbol{C}$を表すに対して，各 $T_i \in \boldsymbol{T}$の検索結果による一貫性制約の充足可能性問題と変更結果に対する一貫性検証問題． □

論理性に関する問題については与えられた H によって結論が一意に決められるが，一貫性に関する問題については一貫性制約と各処理単位の操作結果によって結論が異なる．本論文では，論理性に関する問題に焦点を当てる．ただし，論理性に関する問題に対して，それを判定する問題もNP完全である[11]．このため，D-直列可能性と同様に，D-等価による部分クラス $L_D(\boldsymbol{T})$ を用いる．

定義10[11] $L_D(\boldsymbol{T})$ 判定グラフ $DG(H)$ は，各 $T_i \in \boldsymbol{T}$内の操作 $A_i(X)(A \in \{R, W\})$ を節点とし，枝は次の条件を満たすときに作られる．

- 処理単位内枝：$A_i(X) <_i A_i(Y)$ ならば，$A_i(X)$ から $A_i(Y)$ への枝を持つ．
- 処理単位間R枝：$W_i(X) <_H R_j(Y)(X \cap Y \neq \phi, i \neq j)$ ならば，$W_i(X)$ から $R_j(Y)$ への枝を持つ．
- 処理単位間W枝：$A_i(X) <_H W_j(Y)(X \cap Y \neq \phi, i \neq j)$ ならば，T_i 内のすべての操作から $W_j(Y)$ への枝を持つ． □

定理3[11] 与えられた$\boldsymbol{T}$上の H に対して，$L_D(\boldsymbol{T})$ 判定グラフ $DG(H)$ に閉路が存在しなければ，各 $T_i \in \boldsymbol{T}$は論理性を満たすような $H' \sim^* H$ が存在する． □

以降，$L_D(\boldsymbol{T})$ 判定グラフ $DG(H)$ に閉路が存在しないスケジュールの集合を，$L_D(\boldsymbol{T})$ で表す．

次に，D-直列可能ではないが，$L_D(\boldsymbol{T})$ には属するスケジュールの例を示す．

例1 $X \cap Y = X \cap Z = Y \cap Z = \phi$ である次のようなスケジュール H_1 と H_2 について，その $L_D(\boldsymbol{T})$ 判定グラフ $DG(H_1)$ と $DG(H_2)$ は**図1**となる．ただし，

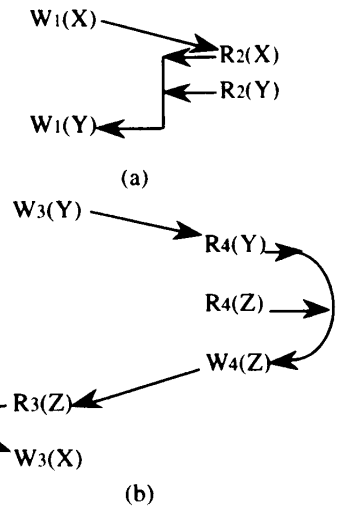

図1 $L_D(\boldsymbol{T})$ クラスに対する判定グラフ $DG(H_1)$，$DG(H_2)$
Fig. 1 Decision graphs $DG(H_1)$ and $DG(H_2)$ for $L_D(\boldsymbol{T})$ class.

H_1 において，X，Y に対する操作は独立に行われると仮定するので，$<_1 = <_2 = \phi$ であり，H_2 において，各処理単位は検索した値のみを変更結果の計算に利用するので，$R_3(Z) <_3 W_3(X)$，$R_4(Y) <_4 W_4(Z)$，$R_4(Z) <_4 W_4(Z)$ である（$<_1, <_2, <_3, <_4$ を全順序としても，全体の結果には影響ない）．$DG(H_1)$ と $DG(H_2)$ のどちらにも閉路は存在しないので，H_1，H_2 ともに $L_D(\boldsymbol{T})$ に属する．また，$DG(H_1)$ 中の枝 $(R_2(Y), W_1(Y))$，$(R_2(X), W_1(Y))$ のみが処理単位間W枝である．

中間結果参照スケジュール H_1

	T_1	T_2
t_1	$W_1(X)$	
t_2		$R_2(X)$
t_3		$R_2(Y)$
t_4	$W_1(Y)$	

相互参照スケジュール H_2

	T_3	T_4
t_1	$W_3(Y)$	
t_2		$R_4(Y)$
t_3		$R_4(Z)$
t_4		$W_4(Z)$
t_5	$R_3(Z)$	
t_6	$W_3(X)$	

一方，H_1，H_2 のD-直列可能性の判定グラフ $D(H_1)$，$D(H_2)$ は，それぞれ $R_2(Y) <_{H_1} W_1(Y)$ と $W_1(X) <_{H_1} R_2(X)$，$W_3(Y) <_{H_2} R_4(Y)$ と $W_4(Z) <_{H_2} R_3(Z)$ より閉路を持つ．このため，H_1，

H_2 ともにD-直列可能でない．したがって，H_1，H_2 はD-直列可能ではないが，$L_D(\boldsymbol{T})$ クラスには属するスケジュールである． □

H_1 と H_2 に対して，2PL 方式を用いる場合にはそれぞれ $R_2(X)$ と $R_4(Y)$ が T_1 と T_3 が終了するまで待たされることになるので，H_1，H_2 は2PL スケジュールでない．TO 方式を用いる場合にはそれぞれ $R_2(Y) <_{H_1} W_1(Y)$ と $W_4(Z) <_{H_2} R_3(Z)$ に関する条件 $ts(T_2) < ts(T_1)$ と $ts(T_4) < ts(T_3)$ を満たさないことにより，T_1 と T_3 が後退復帰されることになり，H_1，H_2 はTO スケジュールでもない．GT 方式を用いる場合には，D-直列可能性の判定グラフ $D(H_1)$，$D(H_2)$ に閉路を持つことにより，その実行が許可されないことになる．しかし，H_1，H_2 のような直列可能でないスケジュールは，協同作業などの処理に必要とされている[3),6)]．特に，H_1 は文献3) において直列可能性を拡張する必要性を示すために用いられているものである．

4. 2相施錠方式による比較

本章では，$L_D(\boldsymbol{T})$ クラスに対する2相施錠（2PL）方式の変更版（TOE）を導入し，TO と TOE の両スケジュールの性質を比較する．

2PL 方式は，競合関係と解錠の時刻によって定められているので，その変更版もその2点について検討する．まず，各処理単位によって操作されたデータ項目はその処理単位が終了するまで他の処理単位によって変更されていないという論理性の定義により，すでに施錠されるデータ項目に専有施錠はできないが，すでに専有施錠されたデータ項目に共有施錠はできることが分かる．

次に，解錠時刻の変更が可能かどうかについて考察する．

例2 次のスケジュール H_3 の $L_D(\boldsymbol{T})$ 判定グラフ $DG(H_3)$ には，$W_5(Y)\,W_6(X)W_5(Y)$ という閉路が存在する．しかし，t_3 と t_4 より前に，$R_5(X)$ と $R_6(Y)$ のための共有施錠が解錠されたなら，$W_6(X)$ と $W_5(Y)$ は実行できることになる．

	T_5	T_6
t_1	$R_5(X)$	
t_2		$R_6(Y)$
t_3		$W_6(X)$
t_4	$W_5(Y)$	

□

すなわち，施錠と解錠を2相に分けなければ，$L_D(\boldsymbol{T})$ に属さないスケジュール H_3 まで許可される．このため，解錠時刻は変更できない．これらをまとめると，$L_D(\boldsymbol{T})$ クラスのための2相施錠方式の変更版を定義できる．

定義11 2相施錠の変更版（2PLE）方式とは，各 $R_i(X) \in H$ または $W_i(X) \in H$ を行う前に X 中の各要素に対して共有施錠または専有施錠を行い，かつ各処理単位 $T_i \in \boldsymbol{T}$ は施錠部分と解錠部分とに分かれる並行処理制御方式である．ただし，すでに施錠されているデータ項目には他の処理単位が再び専有施錠することはない．2PLE 方式に従うスケジュール H を2PLE スケジュールという． □

例1のスケジュール H_1，H_2 とも2PLE スケジュールであることが分かる．

2PLE スケジュール H は $L_D(\boldsymbol{T})$ に属することを証明するために，$L_D(\boldsymbol{T})$ の判定グラフ $DG(H)$ の性質を分析する．まず，定義10から，未終了の処理単位内の操作を始点とする処理単位間 W 枝のみが始点の作成時刻が終点の作成時刻より遅いことに相当することが分かる．すなわち，このような時間順と逆の枝のみは終点が生成された時点で定義され，始点が生成されるときに確定されるが，他の時間順に沿った枝はすべて終点が生成される時点で確定される．たとえば，例2の判定グラフ $DG(H_3)$ 中の処理単位間 W 枝 $(W_5(Y), W_6(X))$ は時間順と逆の枝である．次に，節点 $A_j(Y)(A \in \{R, W\})$ が生成されるときに，閉路 $A_j(Y), A_k(Z), \cdots, A_i(X), A_j(Y)$ が生じるなら，閉路上の節点の中で $A_j(Y)$ の作成時刻が一番遅いことになる．したがって，枝 $(A_j(Y), A_k(Z))$ は時間順と逆の枝で，$(A_i(X), A_j(Y))$ は時間順に沿った枝である．処理単位間 W 枝のみが時間順と逆の可能性があるため，閉路中の作成時刻が一番遅い節点 $A_j(Y)$ を閉路の始点とすると，次の結論が得られる．

補題1 $L_D(\boldsymbol{T})$ 判定グラフ $DG(H)$ に閉路があれば，その最初の枝は時間順と逆の処理単位間 W 枝である． □

たとえば，例2の判定グラフ $DG(H_3)$ 中の閉路 $W_5(Y)W_6(X)W_5(Y)$ に対して，最初の枝 $(W_5(Y), W_6(X))$ は時間順と逆の処理単位間 W 枝であり，枝 $(W_6(X), W_5(Y))$ は時間順に沿った処理単位間 W 枝である．この結果より，次のことが証明できる．

定理4 2PLE スケジュール H は，$H \in L_D(\boldsymbol{T})$ である． □

証明：$H \notin L_D(\boldsymbol{T})$ とすると，判定グラフ $DG(H)$ に閉路が存在することになる．補題1により，閉路の最初の枝は $A_i(X)$ から変更操作 $W_j(Y)$ への時間順と逆の処理単位間 W 枝である．それは T_i に

$A_i(Z) <_H W_j(Y)(Z \cap Y \neq \phi)$ という $W_j(Y)$ と競合する操作 $A_i(Z)$ が存在することを意味する．すなわち，T_j が T_i の施錠しているデータ項目を変更しているので，$W_j(Y)$ が実行される時点では T_i は解錠部分にある．しかし，$A_i(X)$ から $W_j(Y)$ への時間順と逆の枝は，$W_j(Y)$ が実行される時点で $A_i(X)$ がまだ実行されておらず，T_i は施錠部分にあることを示すため矛盾である． □

最後に，2PLE スケジュールと 2PL スケジュールを比較する．2PLE スケジュールは 2PL スケジュールから専有施錠とその後に施錠される共有施錠間の競合関係をなくしているため，2PL スケジュールは 2PLE スケジュールでもある．一方，例 1 の H_1 と H_2 は 2PL スケジュールではないが，2PLE スケジュールであるため，次の結果が成り立つ．

定理 5 2PL スケジュールの集合は，2PLE スケジュールの集合の真部分集合である． □

すなわち，2PLE 方式では，2PL 方式より専有施錠とその後に施錠される共有施錠間の競合関係をなくしていることにより，検索操作に対しては長時間施錠による遅延問題を解決できた．

5. 時刻印方式による比較

本章では，$L_D(\boldsymbol{T})$ クラスに対する時刻印方式（TO）の変更版（TOE）を導入し，TO と TOE の両スケジュールの性質を比較する．

2 章で示したように，TO 方式とは，処理単位の到着順に時刻印を付け，この時刻印の順番の直列スケジュールと等価となるスケジュールを許可するような方式である．すなわち，処理単位 $T_k \in \boldsymbol{T}$の到着順に時刻印 $ts(T_k)$ を付け，$A_i(X) <_H A_j(Y)$ $(X \cap Y \neq \phi)$ である各競合操作組 $A_i(X)$ と $A_j(Y)$ に対して，$ts(T_i) < ts(T_j)$ であることを要求する方式である．$L_D(\boldsymbol{T})$ クラスのための TO 方式の変更版も，処理単位 $T_k \in \boldsymbol{T}$の到着順に時刻印 $ts(T_k)$ を付け，$A_i(X) <_H W_j(Y)$ である各競合操作組 $A_i(X)$ と $W_j(Y)$ に対して $ts(T_i) < ts(T_j)$ である $H \in L_D(\boldsymbol{T})$ スケジュール H と等価となるスケジュールを許可するような方式となる．しかし，TO 方式では各競合操作組 $A_i(X) <_H A_j(Y)$ に対して，$ts(T_i) < ts(T_j)$ を要求すれば D-直列可能性も保証できたが，その変更版は $A_i(X) <_H W_j(Y)(X \cap Y \neq \phi)$ に対して $ts(T_i) < ts(T_j)$ を要求するだけでは，$H \in L_D(\boldsymbol{T})$ を保証できない．

例 3 次の $X \cap Y = \phi$，$R_9(X) <_9 W_9(Y)$ であるスケジュール H_4 の判定グラフ $DG(H_4)$ には，$R_7(Y)W_8(X)R_9(X)W_9(Y)R_7(Y)$ という閉路が存在するので，$H_4 \notin L_D(\boldsymbol{T})$ である．

	T_7	T_8	T_9
t_1	$R_7(X)$		
t_2		$R_8(Z)$	
t_3		$W_8(X)$	
t_4			$R_9(X)$
t_5			$W_9(Y)$
t_6	$R_7(Y)$		

しかし，処理単位は T_7, T_8, T_9 という順で到着するので，$ts(T_7) < ts(T_8) < ts(T_9)$ である．すなわち，$A_i(X) <_H W_j(Y)$ である競合操作組による順序関係 $R_7(X) <_{H_4} W_8(X)$ に対する時刻印の順序 $ts(T_7) < ts(T_8)$ は満たすが，$H_4 \notin L_D(\boldsymbol{T})$ である． □

ここで，例 3 の $DG(H_4)$ 中の閉路 $R_7(Y)W_8(X)R_9(X)W_9(Y)R_7(Y)$ を分析してみる．この閉路は，時間順と逆の処理単位間 W 枝 $(R_7(Y), W_8(X))$ を最初の枝とする経路によって構成される．枝 $(R_7(Y), W_8(X))$ は，$R_7(X) <_{H_4} W_8(X)$ によって生成されている．このため，TO 方式の変更版は，次のような考えで各操作にも時刻印を設け，閉路を排除する．$A_i(X) <_H W_j(Y)$ $(i \neq j, X \cap Y \neq \phi)$ の場合に，T_i のすべての操作から $W_j(Y)$ への処理単位間 W 枝を記述するために，$ts(T_i)$ を $W_j(Y)$ の時刻印として記録する．そして，それを最初の枝とする経路が延長されるときに，$W_j(Y)$ の時刻印をその後の経路上の節点の時刻印ともなるようにする．ただし，複数の経路が 1 つの節点を通る場合には，大きい方の時刻印をとる．すなわち，各操作の時刻印はその操作を通る処理単位間 W 枝を最初の枝とする経路情報を記述する．例 3 のように検索操作時に生じる閉路は，$W_i(X) <_H R_j(Y)(i \neq j, X \cap Y \neq \phi)$ に対する条件 $ts(W_i(X)) < ts(T_j)$ を設けることにより排除する．

定義 12 時刻印方式の変更版（TOE）方式とは，各処理単位 $T_i \in \boldsymbol{T}$の到着順に時刻印 $ts(T_i)$ (> 0) を割り当て，次のように各操作 $A_i(X) \in H$ の時刻印 $ts(A_i(X))$ を計算し，各種の時刻印間の条件を満たすように制御する並行処理制御方式である．ただし，$ts(A_i(X))$ の初期値は 0 である．TOE 方式に従うスケジュール H を TOE スケジュールという．

(1) $A_i(X) <_i A_i(Y)$ の場合：
計算：
$ts(A_i(Y)) := MAX(ts(A_i(Y)), ts(A_i(X)))$

(2) $W_i(X) <_H R_j(Y)$ $(i \neq j,\ X \cap Y \neq \phi)$ の場合：

条件： $ts(W_i(X)) < ts(T_j)$

計算：

$ts(R_j(Y)) := MAX(ts(R_j(Y)), ts(W_i(X)))$

(3) $A_i(X) <_H W_j(Y)$ $(i \neq j,\ X \cap Y \neq \phi)$ の場合：

条件： $ts(T_i) < ts(T_j)$

計算：

$ts(W_j(Y)) := MAX(ts(W_j(Y)), ts(T_i))$ □

すなわち，$W_i(X) <_H R_j(Y)(i \neq j,\ X \cap Y \neq \phi)$ に対しては，$ts(W_i(X)) < ts(T_j)$ という条件を設けている．例3の $L_D(\boldsymbol{T})$ クラスに属さない H_4 に対して，$ts(T_7) = 1$，$ts(T_8) = 2$，$ts(T_9) = 3$ とすると，$R_7(X) <_{H_4} W_8(X)$ の場合，$ts(W_8(X)) = ts(T_7) = 1$ となり，それが $R_9(X)$，$W_9(Y)$ の時刻印にもなる．すなわち，$ts(W_8(X)) = ts(R_9(X)) = ts(W_9(Y)) = 1$ である．$W_9(Y) <_{H_4} R_7(Y)$ の処理時に $ts(W_9(Y)) < ts(T_7)$ を満たさないので，H_4 はTOEスケジュールでない．

例4 例1の H_2 と $H_1' \sim^* H_1$ である次の $H_1' \in L_D(\boldsymbol{T})$ について考察する．

	T_1	T_2
t_1		$R_2(Y)$
t_2	$W_1(X)$	
t_3		$R_2(X)$
t_4	$W_1(Y)$	

H_1' に対して，$ts(T_2) = 1$，$ts(T_1) = 2$ とすると，t_3 時点で $W_1(X) <_{H_1'} R_2(X)$ に対して $ts(W_1(X)) < ts(T_2)$ かどうかが検査される．$ts(W_1(X)) = 0$ のため，条件を満たす．t_4 の時点で $R_2(Y) <_{H_1'} W_1(Y)$ に対して $ts(T_2) < ts(T_1)$ かどうかが検査される．こちらも条件を満たすので，H_1' はTOEスケジュールである．

一方，H_2 に対して，$ts(T_3) = 1$，$ts(T_4) = 2$ とすると，t_2 時点で $W_3(Y) <_{H_2} R_4(Y)$ に対する $ts(W_3(Y)) < ts(T_4)$ と，t_5 時点で $W_4(Z) <_{H_2} R_3(Z)$ に対する $ts(W_4(Z)) < ts(T_3)$ が検査される．$ts(W_3(Y)) = ts(W_4(Z)) = 0$ のため，どちらの条件も満たし，H_2 もTOEスケジュールである． □

次に，TOEスケジュールは $L_D(\boldsymbol{T})$ クラスに属することを証明する．

補題2 TOEスケジュールにおいて，各 $A_i(X) \in T_i \in \boldsymbol{T}$ に対して $ts(A_i(X)) < ts(T_i)$ である． □

証明：定義12において，(2)と(3)で，各操作には所属する処理単位の時刻印より小さい時刻印が与えられており，(1)は同じ処理単位内の操作間で時刻印の引き渡しを行っている．このため，補題が成立する． □

定理6 TOEスケジュール H は，$H \in L_D(\boldsymbol{T})$ である． □

証明：補題1により，$DG(H)$ グラフにおける閉路は，時間順と逆の処理単位間W枝を最初の枝とする経路によって構成される．定義12により，$DG(H)$ において処理単位間W枝を生成させる $A_i(X) <_H W_j(Y)$ $(A \in \{R, W\}, i \neq j, X \cap Y \neq \phi)$ に対し，T_i の時刻印が T_j の時刻印より小さいなら，T_i の時刻印 $ts(T_i)$ は $W_j(Y)$ の時刻印となる．その処理単位間W枝の他の種類の枝による延長時には，$W_j(Y)$ の時刻印は経路上のその後の操作の属する処理単位の時刻印より小さいことが要求される．その要求が満たされれば，$W_j(Y)$ の時刻印 $(= ts(T_i))$ は経路上のその後の操作の時刻印となる．したがって，そのような経路の枝 $(W_k(X), R_i(Z))(k \neq i)$ による閉路は，$W_k(X) <_H R_i(Z)$ に対して，$ts(W_k(X)) = ts(T_i)$ より条件 $ts(W_k(X)) < ts(T_i)$ を満たさないため排除される．

一方，経路の処理単位間W枝 $(A_k(X), W_n(Z))$ による延長は，$ts(T_k) < ts(T_n)$ 時にしか許可されない．$ts(A_k(X)) = ts(T_i)$，$ts(T_k) > ts(A_k(X))$（補題2）より，$ts(T_k) > ts(T_i)$ となるため，$W_n(Z) = W_i(Z)$ という閉路になる場合は，まず排除される．T_k の時刻印は T_i の時刻印より大きいことから，$W_n(Z)$ およびその後の経路上の操作の時刻印は T_i の時刻印より大きいことが保証される．したがって，その後の枝延長時に生じる閉路は，枝 $(A_l(U), A_i(V))$ の生成時に T_i の時刻印が $A_l(U)$ の時刻印より小さいことにより排除される．このため，定理が成り立つ． □

最後に，TOスケジュールとTOEスケジュールを比較する．3章で述べたように H_2 はTOスケジュールでない．H_1' も $W_1(X) <_{H_1'} R_2(X)$ に対する条件 $ts(T_1) < ts(T_2)$ を満たさないことにより，TOスケジュールでない．

定理7 TOスケジュールの集合は，TOEスケジュールの集合の真部分集合である． □

証明：両方式は各処理単位の時刻印の付け方が同じであるため，TOスケジュールは，TOEスケジュールの(3)の条件を満たす．(2)の条件も $ts(W_i(X)) < ts(T_i) < ts(T_j)$ により満たされる．したがって，TOスケジュールは，TOEスケジュールである．一方，TOスケジュールでない H_1' や H_2 はTOEスケジュールであることから，定理が成り立つ． □

すなわち，TOE方式は各操作 $A_i(X)$ に対して，$ts(A_i(X)) < ts(T_i)$ である時刻印 $ts(A_i(X))$ を設

け，$W_i(X) <_H R_j(Y)$ に対する検証条件を，TO 方式の $ts(T_i) < ts(T_j)$ から $ts(W_i(X)) < ts(T_j)$ に弱めている．したがって，検索操作による後退復帰の割合もその分減るので，長時間処理結果の紛失問題もその分緩和できる．ここで，各 $T_i \in \boldsymbol{T}$に対して $<_i = \phi$ の場合を用いて，両方式の $W_i(X) <_H R_j(Y)$ に対する検証条件の違いを直観的に示す．定義 12 により，$p <_H W_i(X)$ である $W_i(X)$ との競合操作 p が存在すれば，両方式においては，次のような競合操作組 $A_k(Z) <_H W_i(X)$, $W_i(X) <_H R_j(Y)$ が存在する．$A_k(Z) <_H W_i(X)$, $W_i(X) <_H R_j(Y)$ に対して，TO 方式では，$ts(T_k) < ts(T_i) < ts(T_j)$ を要求するが，TOE 方式では，$ts(T_k) < ts(T_i)$, $ts(T_k)(= ts(W_i(X))) < ts(T_j)$ を要求する．

6. 判定グラフによる比較

本章では，D-直列可能クラスと $L_D(\boldsymbol{T})$ クラスを判定グラフにより比較する．前者の判定グラフ $D(H)$ は処理単位を節点とし，後者の判定グラフ $DG(H)$ は操作を節点としているため，前者の操作を節点とする判定グラフを導入する．

定義 13 $\boldsymbol{T}$上のスケジュール H に対して，D-直列可能クラスの判定グラフ $D_A(H)$ は，各 $T_i \in \boldsymbol{T}$ 内の操作を節点とし，枝は次の条件を満たすときに作られる．

(1) $A_i(X) <_i A_i(Y)$ ならば，$A_i(X)$ から $A_i(Y)$ への枝を持つ．

(2) $W_i(X) <_H R_j(Y)(X \cap Y \neq \phi,\ i \neq j)$ ならば T_i 内のすべての操作から $R_j(Y)$ への枝を持つ．

(3) $A_i(X) <_H W_j(Y)(X \cap Y \neq \phi,\ i \neq j)$ ならば，T_i 内のすべての操作から $W_j(Y)$ への枝を持つ． □

図 2 は，例 1 のスケジュール H_1 と H_2 に対する判定グラフ $D_A(H_1)$ と $D_A(H_2)$ を示している．これは図 1 の $DG(H_1)$ に枝 $(W_1(Y), R_2(X))$, $DG(H_2)$ に枝 $(R_3(Z), R_4(Y))$, $(W_3(X), R_4(Y))$, $(R_4(Y), R_3(Z))$, $(R_4(Z), R_3(Z))$ を追加したものである．$W_1(X) <_{H_1} R_2(X)$, $R_2(Y) <_{H_1} W_1(Y)$ より，$D_A(H_1)$ では $W_1(Y)$ と $R_2(X)$ 間に閉路が生じる．$W_3(Y) <_{H_2} R_4(Y)$, $W_4(Z) <_{H_2} R_3(Z)$ より，$D_A(H_2)$ には $R_3(Z)$ と $R_4(Y)$ 間に閉路を持つことになる．

D-直列可能性の 2 種類の判定グラフ間には，次の関係がある．

補題 3 $\boldsymbol{T}$上のスケジュール H に対して，判定グラフ $D_A(H)$ に閉路が存在するための必要十分条件は，判定グラフ $D(H)$ に閉路が存在することである． □

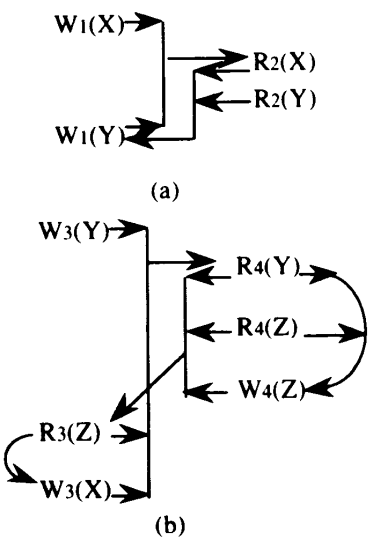

図 2 D-直列可能クラスに対する判定グラフ $D_A(H_1)$, $D_A(H_2)$
Fig. 2 Decision graphs $D_A(H_1)$ and $D_A(H_2)$ for D-serializable class.

必要性：判定グラフ $D_A(H)$ において同一の処理単位 T_i 内の操作間は，$A_i(X) <_i A_i(Y)$ 時にのみ $A_i(X)$ から $A_i(Y)$ への枝を持つ．$<_i$ は半順序であるため，まず，同一の処理単位内の操作間には閉路は存在しえない．次に，$D(H)$ は $D_A(H)$ 内の同一の処理単位に属する操作を 1 つの節点にしたグラフであることが確認できる．このため，判定グラフ $D_A(H)$ に閉路が存在すれば，D-直列可能性判定グラフ $D(H)$ にも必ず閉路が存在することになる．

十分性：$D(H)$ に $T_{i1}, \cdots, T_{im}, T_{i1}$ という閉路が存在するなら，T_{ij} と $T_{ij+1}(1 \leq j \leq m-1)$ に競合する操作組 $(A_{ij}(X_j), A_{ij+1}(X_{j+1}))(1 \leq j \leq m-1)$, T_{im} と T_{i1} に競合する操作組 $(A_{im}(Y_m), A_{i1}(Y_1))$ が存在することになる．したがって，定義 13 により判定グラフ $D_A(H)$ にも $A_{i1}(Y_1)A_{i2}(X_2), \cdots, A_{im}(X_m)A_{i1}(Y_1)$ という閉路が存在することになる． □

すなわち，$D_A(H)$ はD-直列可能クラスの操作を節点とする判定グラフである．$D_A(H)$ と $DG(H)$ の比較により，次の結果が得られる．

定理 8 D-直列可能クラスは，$L_D(\boldsymbol{T})$ クラスの真部分集合である． □

証明：D-直列可能クラスの判定グラフ $D_A(H)$ と $L_D(\boldsymbol{T})$ クラスの判定グラフ $DG(H)$ は，同じ節点集合を持つが，$W_i(X) <_H R_j(Y)$ $(X \cap Y \neq \phi)$ の場合に，$D_A(H)$ の方がより多くの枝を持つ．したがっ

て，$DG(H)$ に閉路が存在するなら，$D_A(H)$ にも必ず閉路が存在する．さらに，$D_A(H_1)$, $D_A(H_2)$ には閉路は存在するが，$DG(H_1)$, $DG(H_2)$ には閉路は存在しないという例1の H_1, H_2 より，定理が成り立つ．□

まとめると，$L_D(\boldsymbol{T})$ クラスの判定グラフ $DG(H)$ は D-直列可能クラスの判定グラフ $D_A(H)$ より，$W_i(X) <_H R_j(Y)$ の場合に T_i のすべての操作から $R_j(Y)$ への枝を $W_i(X)$ から $R_j(Y)$ への枝まで減らしている．すなわち，$D_A(H)$ と $DG(H)$ において，各検索操作への平均的な枝数は $(\Sigma_{i=1}^{n} \mid T_i \mid) : n$ になる．ただし，$\mid T_i \mid$ は T_i 内の操作の数で，n は $\boldsymbol{T}$ 内の処理単位の数である．検索操作が実行されるときに，$D_A(H)$ と $DG(H)$ に生じる閉路の数を，検索操作への枝数で概算すると，検索操作による後退復帰は $L_D(\boldsymbol{T})$ クラスがD-直列可能クラスの $(\Sigma_{i=1}^{n} \mid T_i \mid /n)$ 分の1に減少している．

7. おわりに

本論文では，並行処理制御方式の視点から，独立化可能という一貫性情報を持つ高水準DBにおける新たな基準と，従来の直列可能基準との比較を行った．従来のよく知られている2相施錠方式，時刻印方式，グラフ判定方式の変更版を与えたうえ，変更版と元の方式の比較を行った．それらによって，独立化可能クラスは直列可能クラスより検索操作に対して制限が弱められたという結論が得られた．

具体的に，2相施錠方式からは，専有施錠とその後に施錠される共有施錠間の競合関係をなくす拡張ができた．それにより，検索操作に対しては長時間施錠による遅延問題を解決できた．時刻印方式からは，各操作 $A_i(X)$ に対して $ts(A_i(X)) < ts(T_i)$ である時刻印 $ts(A_i(X))$ を設け，$W_i(X) <_H R_j(Y)$ に対する検証条件を $ts(T_i) < ts(T_j)$ から $ts(W_i(X)) < ts(T_j)$ まで弱める拡張ができた．それにより，検索操作による後退復帰もその分減るので，長時間処理結果の紛失問題もその分緩和できる．グラフ判定方式からは，D-直列可能性の判定グラフより，$W_i(X) <_H R_j(Y)$ に対する T_i のすべての操作から $R_j(Y)$ への枝を $W_i(X)$ から $R_j(Y)$ への枝まで減らす拡張ができた．それにより，検索操作による後退復帰がD-直列可能クラスの $(\Sigma_{i=1}^{n} \mid T_i \mid /n)$ 分の1に減少していることになり，H_1, H_2 のような非直列可能である協同作業スケジュールを許可できることになる．

本論文の最も重要な貢献は，独立化可能性という直列可能性に代わる正当性基準の特性を並行処理制御方式の視点からまとめられたことである．それは独立化可能性の必要性と有効性を議論するうえで重要なものとなる．

参考文献

1) Agrawzl, R., Carey, M.L. and Livny, M.: Concurrency Control Performance Modeling: Alternatives and Implications, *ACM Trans. Database Syst.*, Vol.12, No.4, pp.609-654 (1987).
2) Agrawal, D., Abbadi, A.E. and Singh, A.K.: Consistency and Orderability: Semantics-Based Correctness Criteria for Databases, *ACM Trans. Database Syst.*, Vol.18, No.3, pp.460-486 (1993).
3) Barghouti, N.S. and Kaiser, G.E.: Concurrency Control in Advanced Database Applications, *Comput. Surv.*, Vol.23, No.3, pp.269 317 (1991).
4) Bernstein, P.A., Hadzilacos, V. and Goodman, N.: *Concurrency Control and Recovery in Database Systems*, Addison-Wesley (1987).
5) Bernstein, P.A., Shipman, D.W. and Wong, W.S.: Formal Aspects of Serializability in Database Concurrency Control, *IEEE Trans. Softw. Eng.*, Vol.5, No.5, pp.203-216 (1979).
6) Elmagarmid, A.K.: *Database Transaction Models for Advanced Applications*, Morgan Kaufmann (1990).
7) Korth F.: Formal Aspects of Concurrency Control in Long-Duration Transaction Systems Using the NT/PV Model, *ACM Trans. Database Syst.*, Vol.19, No.3, pp.450-491 (1994).
8) Papadimitriou, C.H.: The Serializability of Concurrent Database Updates, *J. ACM*, Vol.26, No.4, pp.631-653 (1979).
9) 上林弥彦：データベースの基礎理論 (6) 共有データベースの諸問題に対する理論，情報処理学会誌，Vol.24, No.8, pp.1020-1037 (1983).
10) 室章治郎：データベースの同時処理制御における直列可能性の理論，情報処理学会誌，Vol.26, No.9, pp.1011-1022 (1985).
11) 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).

(平成7年10月16日受付)
(平成8年5月10日採録)

徐 海燕（正会員）

昭和58年中国復旦大学理学部計算機科学科卒業．平成2年九州大学大学院博士後期課程修了．工学博士．同年福岡工業大学電子工学科講師，現在同助教授．高水準データベースの質問処理論，並行処理制御などの研究に従事．電子情報通信学会，日本ソフトウェア科学会，ACM各会員．

古川 哲也（正会員）

昭和58年京都大学工学部卒業．昭和60年京都大学大学院修士課程修了．昭和63年九州大学大学院博士後期課程修了．工学博士．九州大学工学部助手，同大大型計算機センター講師，同助教授を経て，平成6年九州大学経済学部助教授，現在に至る．この間，平成7年米国パディユ大学客員研究員．データベースの設計論・質問処理論，情報システムなどの研究に従事．電子情報通信学会，ACM，IEEE，日本OR学会等会員．

史 一華

昭和57年中国復旦大学理学部計算機科学科卒業．平成4年九州大学大学院博士後期課程修了．理学博士．同年福岡工業短期大学電子電子情報システム学科助教授．現在に至る．時間限定推論，知識ベースシステム，真理保全などの研究に従事．電子情報通信学会，人工知能学会各会員．