ワークフロートランザクションの隔離性

徐 海燕[†] 古川哲也[††]

今日の情報社会でワークフロー管理システムの果たす役割がますます増えている．複数のワークフローのインスタンスまたは同一ワークフローの異なるインスタンスで検索・更新されるデータ項目に共通のものがあれば，並行処理制御が必要となる．データベースの並行処理制御は，トランザクションの一貫性と隔離性によって実行結果のデータベースの一貫性を保証している．しかし，ワークフロートランザクションの一貫性は，入力データはタスク単位，出力データはトランザクション単位で満足される．タスクは入出力条件で表されるアプリケーションの一貫性情報も利用できる．したがって，隔離性の定義もそれに応じて再検討する必要がある．本論文では，入力データの隔離性と出力データの隔離性を導入し，それらによって並行実行における入出力データの一貫性を保証できることを示す．さらに，入出力条件を用いた隔離性の判定方法について検討し，それらの相互関連について議論する．隔離性を細分することによって許される並行実行は，直列可能でないものや直列実行では生成されることのないものを含む．

Isolation Property of Workflow Transactions

Haiyan Xu[†] and Tetsuya Furukawa[††]

To facilitate business process and information process reengineering and automation, workflow management systems have been widely used in the applications such as e-commerce and manufacturing. Serializability as an isolation criterion for traditional transactions, would overly restrict the concurrency of interleaved execution of workflow instances. Since for workflow transactions the consistency units of output data are transactions while their consistency units of input data are tasks, in this paper, we separate the isolation for input and output data respectively. The mechanisms for the both isolations are developed by using the input and output conditions, i.e., the application semantics. With the mechanisms, we show the difference between our approach and the previous ones.

1. はじめに

ビジネスプロセスの自動化，および情報共有の効率化を図るため，多くの企業においてワークフロー管理システムが導入されている[1)]．その下では，個々の仕事（タスク）と仕事の流れ（制御フロー）を事前に定義し，それに従って順次仕事を処理する．ワークフロー内のいくつかのタスク，または異なるワークフローのタスクの操作するデータが重なることがある．すなわち，ワークフローによるタスクの実行で共有データが存在する．共有データが存在していれば，タスクの実行の正当性を保証するために並行処理制御が必要となる[10)]．

たとえば，注文を受けた後，同時に配達と入金のタスクを実行する注文業務と，受付の後，入金していれば返金し，納品していれば返品するというように順次処理を行うキャンセル業務を考える．ある顧客が注文し入金したのちキャンセルを行ったとき，並行処理制御を行わないと，返金したにもかかわらず，品物は配達してしまうという事態になりかねない．

従来のデータベースの並行処理制御は，トランザクションの原子性，一貫性，隔離性，永続性，すなわちACID特性と呼ばれる性質を保証するために行われている[4)]．単独実行時に各トランザクションが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仮定することで一貫性を，直列可能な並行実行を正当とすることで隔離性を実現している．また，原子性と永続性によって故障に対する回復制御の基準を与えている．

ワークフローにおいては，作業の流れである制御フ

† 福岡工業大学情報工学部情報工学科
Department of Computer Science and Engineering, Fukuoka Institute of Technology
†† 九州大学大学院経済学研究院
Department of Economic Engineering, Kyushu University

ローやデータの流れであるデータフローに関する情報が利用でき，作業の基本単位であるタスクに関しても，入出力条件としてアプリケーションの一貫性情報がある．原子性の単位は，トランザクションからタスクへ短縮されている[12]．一貫性については，入力データはタスク単位で，出力データはトランザクション単位で分けて扱う[3]．このため，一貫性を満たすトランザクションからなるスケジュールにおいて，各トランザクションを一貫したデータベース上の遷移とするためには，隔離性も一貫性の単位に応じて再定義する必要が生じる．

従来の直列可能性に基づく並行処理制御方式では，ワークフローの特徴に対応できていない．このことは広く認識されており，ワークフローに対する並行処理制御は研究なされているが[6)~8),11)]，一般に受け入れられている正当性基準は存在しない[3]．そのため，論文 3) では，ワークフローを形式的に定義し，データベースの一貫性の必要十分条件が一貫性制約として利用できる場合の正当性基準を，(1) 各節点の入力データは入力条件を満たす，(2) 外部データは一貫性制約を満たす，(3) スケジュール終了時でのデータベースは一貫している，としている．しかし，データベースの一貫性の必要十分条件を定義できるアプリケーションは限られているので，タスクの入出力条件のみに基づくワークフロートランザクションの正当性基準も重要である．一方，論文 9) では，ワークフローの設計者が並行実行に対応できるように設計したワークフローに対して，SQL 言語を用いた並行処理制御の実現方法を示している．ここでは各トランザクションの実行ごとに満たすべき条件と期間を設定することを前提としている．

ワークフローモデルでは，タスクの独立性が高いため，タスクの正当な実行のために入出力データに対して条件を設定する．本論文では，そのような条件である一貫性情報のみを利用した場合におけるトランザクションの隔離性について検討する．隔離性は，並行実行時に他のトランザクションによるデータの一貫性に対する影響を受けず，単独実行と同様に正当な実行を保証するための性質である．一貫性が入力データと出力データに対して定義されているのに対応して，隔離性も入力データと出力データに分けて定義することの必要性を指摘し，入力データおよび出力データの隔離性を導入する．入力データの隔離性については，データベースからの入力である外部入力データと先行タスクの出力である内部入力データに分けて検討を行う．内部入力データの隔離性は，出力データの隔離性と密接に関連し，入力条件によって判定できるが，外部入力データの隔離性については，競合関係による新たな判定方法が必要である．出力データの隔離性と内部入力データの隔離性については，それらを保証する十分条件と必要十分条件をそれぞれ提案する．外部入力データの隔離性については，十分条件となる判定グラフ方法を提案する．入出力データの隔離性を満たすスケジュールは，直列可能でないスケジュールだけでなく，直列実行では生成されることのない実際の作業に近いスケジュールも含むことを示す．

本論文は，次のように構成される．2 章でワークフローモデルを定義し，3 章ではそれに基づいて入出力データの隔離性を導入する．出力データと内部入力データの隔離性については 4 章で，外部入力データの隔離性については 5 章で検討する．6 章では本論文の結果についての議論を行い，7 章は全体のまとめである．

2. ワークフローモデル

ワークフローは，タスクとタスク間の実行順序を与える制御フローによって記述される．タスク t はその特徴を表す 4 つのパラメータ，入力/出力データ項目の集合 I/O，入力/出力データに対する条件 IC/OC からなる組 $t(I, O, IC, OC)$ によって記述される[6]．入力条件 IC（出力条件 OC）は，入力データ（出力データ）に対する条件の集合である．同一項目に対する条件は，肩字で入力データと出力データを区別する．

例 1 商品販売のワークフロー管理システムには，次の 4 つのタスクがあるとする．簡単のため，単価が $tanka$ である単一の商品を販売するものとし，配達者も 1 人であるとする．

- 注文 t_1：
  I_1={ 顧客 ID:x_1, 数量:x_2, 在庫数:x_3},
  O_1={ 取寄せ数:x_4},
  $IC_1 = \{x_3 \geq 0,\ x_2 > 0\}$,
  $OC_1 = \{(x_3 + x_4 \geq x_2) \vee (x_3 \geq x_2\}$.
- 入荷 t_2：
  I_2 ={ 取寄せ数:x_4, 数量:x_2, 在庫数:x_3},
  O_2 ={ 在庫数:x_3},
  $IC_2 = \{x_4 > 0\}$,
  $OC_2 = \{x_3^O = x_3^I + x_4\}$.
- 支払 t_3：
  I_3={ 顧客 ID:x_1, 数量:x_2},
  O_3={ 支払額:x_5},
  $IC_3 = \{x_2 > 0\}$,
  $OC_3 = \{x_5 = x_2 * tanka, x_5 > 0\}$.

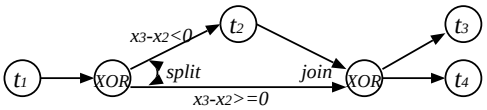

図 1 制御フロー $CF1$
Fig. 1 Control Flow $CF1$.

注文 $\mathcal{T}_1^1$ → 入荷 $\mathcal{T}_2^1$ → 配達 $\mathcal{T}_4^1$ → 支払 $\mathcal{T}_3^1$
(a)
注文 $\mathcal{T}_1^2$ → 支払 $\mathcal{T}_3^2$ → 配達 $\mathcal{T}_4^2$
(b)

図 2 トランザクション WT_1 と WT_2
Fig. 2 Transaction WT_1 and WT_2.

- 配達 t_4：
  I_4={ 数量:x_2, 在庫数:x_3, 配達予定:x_6},
  O_4={ 在庫数:x_3, 配達予定:x_6, 配達数:x_7},
  $IC_4 = \{x_2 > 0, x_3 \geq x_2\}$,
  $OC_4 = \{x_3^O = x_3^I - x_2, x_7 = x_2, x_7 > 0\}$. □

並行実行において，タスク $t(I, O, IC, OC)$ は原子性の単位である．タスク t は入力データ I が入力条件 IC を満たすかどうかを検査する．満たしていれば，生成された出力データ O は，出力条件 OC を満たす．

タスクの実行順序を，タスクを節点とする有向巡回グラフである制御フローによって記述する．順序には，順次，反復，条件分岐（XOR 分岐，XOR 結合）などがある．XOR 分岐にはいくつかの枝分かれがあり，それらの枝にはいずれか 1 つが真となる経路選択条件が記述される．例 1 の注文を受付け，在庫不足の場合のみ取寄せを行い，その後支払と配達を行う制御フロー CF_1 を**図 1** に示している．

トランザクションは，ワークフローに従うタスクの実行列である．形式的に記述するために，データ項目 x およびデータ項目集合 X のインスタンスを $\mathcal{I}_x$ および $\mathcal{I}_X$ とし，$\mathcal{T}(\mathcal{I}_I, \mathcal{I}_O, IC, OC)$ でタスク $t(I, O, IC, OC)$ のプロセスを記述する．トランザクションは，タスクのプロセスを表す節点集合とそれらの間の実行順序を表す枝集合からなる有向グラフ $WT(TN, TE)$ で定義する．$WT(TN, TE)$ において XOR 分岐節点は省略することがあるが，省略しない場合は，真となる経路選択条件がトランザクション内の XOR 分岐節点の出力条件となる．

例 2 図 1 の制御フロー CF_1 からは，いくつかのトランザクションが得られる．$x_3 < x_2$ のとき，入荷が必要となる．**図 2** (a) は入荷後，配達，支払の順で実行されるトランザクション WT_1 である．$x_3 \geq x_2$ のとき取寄せによる入荷は必要としない．図 2 (b) はそのような場合に支払が配達よりも先に行われるトランザクション WT_2 である． □

トランザクションは，全順序グラフである．これは，制御フローが巡回の場合は複数回実行されるタスクは異なるプロセスに展開され，条件分岐は特定の経路のタスクに従って実行されるためである．

最後にトランザクションの並行実行を表すスケジュールを定義する．制御フロー CF_j $(j = 1, 2, \cdots)$ 上のトランザクション $WT_i(TN_i, TE_i)$ $(i = 1, 2, \cdots)$ からなるスケジュールは，次のような有向非巡回グラフ $WH(HN, HE)$ である．

- $HN = \bigcup_i TN_i$
- $HE \supseteq \bigcup_i TE_i$
- 異なる処理単位の節点 $\mathcal{T}_s$ と $\mathcal{T}_t$ が競合する $(O_s \cap (I_t \cup O_t) \neq \phi) \vee (O_t \cap (I_s \cup O_s) \neq \phi))$ なら，それらの間には実行順序を示す枝か推移的な枝が HE に含まれる．

3. トランザクションの隔離性

トランザクションの並行実行では，各トランザクションは単独実行のときと同様に正当な実行でなければならない．本章では，単独実行時に各トランザクションが満たさなければならない性質を一貫性として形式的に定義し，並行実行時に同じ性質を満たすようにするための隔離性について検討する．

ワークフロートランザクションに対して，一貫性は入力データの一貫性と出力データの一貫性に分けられている[3]．入力データの一貫性は，タスク単位で管理しているため，まず，トランザクションの節点間のデータの流れについて考察する．トランザクションにおけるデータの流れは，通常データフローと呼ばれるタスクのプロセスを節点とする有向グラフによって記述される．節点 $\mathcal{T}_i$ がトランザクションの祖先である $\mathcal{T}_j$ の出力データを入力データとして利用するなら，データフローに $\mathcal{T}_j$ から $\mathcal{T}_i$ への枝がある．

各節点の入力データはトランザクション内の他の節点の出力データであることもあり，データベースのデータである場合もある．前者を内部データ，後者を外部データと呼ぶ．このため，入力データの一貫性を，入力データの全体と外部入力データの 2 つの側面から定義する．

定義 1 一貫したデータベースで実行されるスケジュール $WH(HN, HE)$ に対して，各節点 $\mathcal{T} \in HN$

の入力データはその入力条件を満たし，外部入力データは一貫したデータベースのものであるとき，スケジュール $WH(HN, HE)$ は**入力一貫性**を満たすという．□

ワークフローの設計では，通常内部データが入力条件を満たすことはデータフローの親節点の出力条件によって保証され，一貫したデータベースで実行されれば入力条件は満たされると仮定されている．たとえば，WT_1 の配達 $\mathcal{T}_4^1$（WT_2 の配達 $\mathcal{T}_4^2$）の $x_3 \geq x_2$ という入力条件は，入荷 $\mathcal{T}_2^1$（注文 $\mathcal{T}_1^2$）の出力条件によって保証されている．このため，内部データが入力条件を満たすこと，外部データが一貫したデータベースのものであることをそれぞれ内部入力データの一貫性と外部入力データの一貫性という．

一方，出力一貫性は，入力一貫性を満たすトランザクションが一貫したデータベースにおいて単独で実行されたとき，その結果のデータベースは一貫していることを意味する．その形式的な議論にトランザクションの開始から各プロセスまでに使用したデータ項目集合を用いる．

定義 2 トランザクション $WT(TN, TE)$ の実行中のプロセス $\mathcal{T}_i \in TN$ に対し，それまでの各プロセスの入力データ項目と出力データ項目の和集合を $\mathcal{T}_i$ の使用データ項目 D_i といい，それらの最新値を $\mathcal{I}_{D_i}$ で表す．$\mathcal{T}_i$ までの入出力条件で最新値 $\mathcal{I}_{D_i}$ のみに関わるものからなる集合を，$\mathcal{T}_i$ の結果条件 TC_i という．また，トランザクション WT の最終プロセスの使用データ項目，最新値，結果条件を，トランザクション WT の使用データ項目，最新値，結果条件という．□

トランザクションの結果条件は，トランザクション単位で考えたときの出力条件と見なすことができる．

例 3 在庫数が 100 個であるときに顧客 A が 2 個を注文し，12 月 11 日に配達させるトランザクションを WT_2 とする．それぞれ注文 $\mathcal{T}_1$，支払 $\mathcal{T}_3$，配達 $\mathcal{T}_4$ の実行が終了した時点での使用データ項目，最新値，結果条件は，次のようになる．

- $D_1 = \{x_1, x_2, x_3, x_4\}$,
  $\mathcal{I}_{D_1} = \{x_1 = A,\ x_2 = 2,\ x_3 = 100, x_4 = null\}$,
  $TC_1 = \{x_2 > 0,\ x_3 \geq 0,\ x_3 \geq x_2\}$.
- $D_3 = \{x_1, x_2, x_3, x_4, x_5\}$,
  $\mathcal{I}_{D_3} = \{x_1 = A,\ x_2 = 2,\ x_3 = 100,\ x_4 = null,$
  $x_5 = 2 * tanka\}$,
  $TC_3 = \{x_2 > 0,\ x_3 \geq 0,\ x_3 \geq x_2,\ x_5 = 2 * tanka,\ x_5 > 0\}$.
- $D_4 = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7\}$,
  $\mathcal{I}_{D_4} = \{x_1 = A,\ x_2 = 2,\ x_3 = 98,\ x_4 = null,$
  $x_5 = 2 * tanka,\ x_6 = 02/12/11,\ x_7 = 2\}$,
  $TC_4 = \{x_2 > 0,\ x_3 = 98,\ x_5 > 0,\ x_7 = x_2,\ x_7 > 0\}$. □

トランザクションの実行に対するデータベースの一貫性は，各タスクの出力条件によって保持される，すなわち，トランザクションの単独実行ですべてのタスクが出力条件を満たすデータを出力すれば，データベースは一貫していると仮定する．これによって出力一貫性を形式的に定義できる．

定義 3 一貫したデータベースにおいて単独で実行されるトランザクション $WT(TN, TE)$ に対して，WT の終了時に，WT の使用データ項目の最新値がトランザクションの結果条件を満たすならば，WT は**出力一貫性**を満たしているという．□

一貫したデータベースでトランザクションが単独実行されるとき，トランザクションが入力一貫性と出力一貫性を満たせば実行結果のデータベースも一貫している．しかし，並行実行においては，出力一貫性を満たすトランザクションからなるスケジュールに対して入力一貫性を保証するだけでは，スケジュール終了後のデータベースの一貫性を保証できない．

例 4 例 1 の会社に注文のキャンセルを扱うワークフローもあり，それには次の 3 つのタスクがあるとする．

- 受付 t_5：
  I_5={ 顧客 ID:x_1, 数量:x_2},
  O_5=ϕ,
  $IC_5 = \{x_2 > 0\}$,
  $OC_5 = \phi$.
- 返金 t_6：
  I_6={ 顧客 ID:x_1, 支払額:x_5},
  O_6={ 返金額:x_8, 支払額:x_5},
  $IC_6 = \{x_5 > 0\}$,
  $OC_6 = \{x_8 = x_5^I,\ x_5^O = x_5^I - x_8\}$.
- 返品 t_7：
  I_7={ 顧客 ID:x_1, 配達数:x_7, 在庫数:x_3},
  O_7={ 返品数:x_9, 在庫数:x_3},
  $IC_7 = \{x_7 > 0\}$,
  $OC_7 = \{x_9 = x_7,\ x_3^O = x_3^I + x_9\}$.

制御フロー CF_2 は，受付の後，入金していれば返金し，納品していれば返品するというように順次処理を行うとする（**図 3**）．

図 4 は，WT_2 で注文し，入金したのち，キャンセルトランザクション WT_3 が実行されるスケジュール WH_1 である．$\mathcal{T}_6^3$ は支払 $\mathcal{T}_3^2$ に対する返金なの

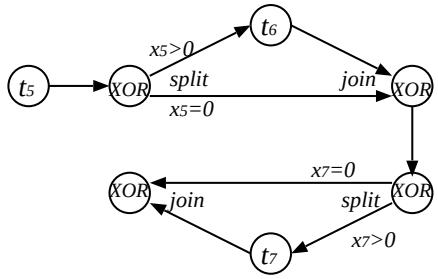

図 3 制御フロー $CF2$
Fig. 3 Control Flow $CF2$.

注文 $\mathcal{T}_1^2 \rightarrow$ 支払 $\mathcal{T}_3^2 \rightarrow$ 配達 $\mathcal{T}_4^2$
$\downarrow \quad \uparrow$
受付 $\mathcal{T}_5^3 \rightarrow$ 返金 $\mathcal{T}_6^3 \rightarrow \mathrm{XOR}^3$

図 4 出力データの隔離性を満たさない WH_1
Fig. 4 WH_1 not satisfying isolation of output data.

で，$\mathcal{T}_3^2$ から $\mathcal{T}_6^3$ への枝がある．一方，XOR 分岐節点 XOR^3 は配達 $\mathcal{T}_4^2$ より前に実行され，配達数 x_7 を検索したため，XOR^3 から $\mathcal{T}_4^2$ への枝がある．すなわち，返金したにもかかわらず品物は出荷しており，実行結果のデータベースは一貫していない． □

並行実行におけるトランザクションの入力一貫性と実行結果のデータベースの一貫性を保証するために，それぞれ入力データの隔離性と出力データの隔離性を定義する．

定義 4 トランザクション $WT_i(TN_i,TE_i)$ $(i = 1, 2, \cdots)$ からなるスケジュール $WH(HN, HE)$ に対して，次の 2 つの条件をトランザクション WT_i に対する入力データの隔離性および出力データの隔離性という．

- **入力データの隔離性**：$WT_i(TN_i,TE_i)$ 中の各節点 $\mathcal{T}_s \in TN_i$ が，他のトランザクションの並行実行によって入力一貫性を満たさなくなることが生じない．
- **出力データの隔離性**：各 WT_i が他のトランザクションの並行実行によって出力一貫性を満たさなくなることが生じない． □

たとえば，WH_1 において，WT_2 の終了時の使用データ項目 x_5 のデータベースにおける値が結果条件中の $x_5 > 0$ を満たさないので，出力データの隔離性を満たさない．

入力一貫性が内部入力データの一貫性と外部入力データの一貫性に分けているのに対応して，入力データの隔離性もそれぞれの一貫性を保証するための内部入力データの隔離性と外部入力データの隔離性に分ける．

定理 1 一貫したデータベースで実行される出力一貫性と入出力データの隔離性を満たすトランザクション $WT_i(TN_i, TE_i)$ $(i = 1, 2, \cdots)$ からなるスケジュール $WH(HN, HE)$ において，各節点 $\mathcal{T}_s \in HN$ は入力一貫性を満たし，WH の実行結果のデータベースも一貫している． □

証明． まず，並行実行においても各節点 $\mathcal{T}_s \in HN$ が入力一貫性を満たすことを，データフローの経路の長さ j に対する帰納法で証明する．

- $j = 0$ のとき：節点 $\mathcal{T}_s$ には入る枝がなく，すべての入力データは外部データである．外部入力データは，外部入力データの隔離性より，他のトランザクションの一貫していない中間結果を検索していない．すなわち，他のトランザクションの最終結果か一貫性を満たしている中間結果を検索している．前者の一貫性は，トランザクションの出力一貫性によって保証されている．
- $j = n - 1$ までの場合に成り立つと仮定する．
- $j = n$ のとき：節点 $\mathcal{T}_s$ の内部入力データは，データフローの親節点の出力データである．親節点の経路の長さは $n - 1$ 以下なので，親節点の入力データが入力条件を満たすことは帰納法によって保証される．入力条件を満たせば，親節点の出力データは出力条件を満たす．内部入力データの隔離性より，内部データは入力条件，すなわち，内部入力データの一貫性を満たす．一方，外部データは，外部入力データの隔離性より外部入力データの一貫性を満たす．したがって，節点 $\mathcal{T}_s$ は入力一貫性を満たす．

次に，スケジュール終了時のデータベースも一貫していることを証明する．実行結果のデータベースに一貫していない部分 U があるとする．U が 1 つのトランザクション WT による変更結果であれば，トランザクション WT の出力データの隔離性より，WT の終了時にデータベースにおける WT の使用データ項目の値は WT の結果条件を満たす．WT の出力一貫性より，WT によって変更された部分のデータベースは一貫しており，仮定と矛盾する．複数のトランザクションによる変更結果であれば，非巡回となるスケジュールのグラフ $WH(HN, HE)$ において，それらのトランザクションの節点の枝順での極大節点が存在する．U が一貫していないなら，その極大節点の所属するトランザクションが出力データの隔離性を満たさ

ないことになり，矛盾する． □

4. 出力データと内部入力データの隔離性

出力データの隔離性は，トランザクションの使用データ項目が他のトランザクションの変更によって結果データベースの一貫性を満たさなくならないことを意味する．内部入力データの隔離性は，内部データが入力条件を満たすことを意味する．通常内部データが入力条件を満たすことはデータフローの親節点の出力条件によって保証されるので，両者は密接に関連する．本章では，出力データの隔離性と内部入力データの隔離性の実現方法をあわせて検討する．

定理 2 トランザクション $WT_i(TN_i,TE_i)$ $(i=1, 2, \cdots)$ からなるスケジュール $WH(HN, HE)$ において，WT_i の各実行時点で使用データ項目のデータベースにおける値が直前に終了したプロセスの結果条件を満たせば，WT_i は出力データと内部入力データの隔離性を満たす． □

証明． WT_i の各実行時点で使用データ項目のデータベースにおける値が結果条件を満たすことは，WT_i の終了時点でもトランザクションの使用データ項目のデータベースにおける値がトランザクションの結果条件を満たすことを意味する．定義 4 より，WT_i は出力データの隔離性を満たす．

一方，WT_i は各時点で使用データ項目のデータベースにおける値が結果条件を満たすので，WT_i の各プロセスの開始時点においても結果条件を満たす．データフローの枝が制御フローの経路に対応し，データフローの親節点の出力条件を満たせば子節点の入力条件は保証されるので，内部入力データの隔離性を満たす． □

たとえば，出力データの隔離性を満たさない WT_2 を含む WH_1 において，支払済みを意味する条件 $x_5 > 0$ は，例 3 で示したように支払 $\mathcal{T}_3^2$ の実行終了後から WT_2 の終了まで WT_2 の結果条件に含まれる．返金 $\mathcal{T}_6^3$ はそれが満たされなくなったので，定理 2 を満たさない．

例 5 **図 5** には取寄せを必要としない 2 つのトランザクション WT_2 と WT_4 からなるスケジュール WH_2 が示されている．WT_2 が先に配達スケジュールを決めたため，$\mathcal{T}_4^2$ から $\mathcal{T}_4^4$ への枝がある．スケジュール WH_2 が実行される前の在庫数を 100 とし，WT_2 と WT_4 がそれぞれ 20 個，30 個を注文するとすると，WT_2 の配達 $\mathcal{T}_4^2$ の実行後も WT_4 の注文 $\mathcal{T}_1^4$ の出力条件 $x_3 \geq 30$（在庫数は注文数以上）は保たれている．このため，在庫数 100 の場合における WH_2 の実行は，定理 2 を満たす． □

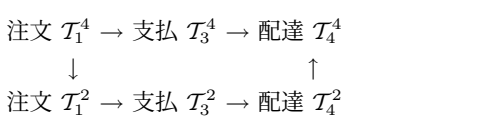

図 5 定理 2 を満たすスケジュール WH_2
Fig. 5 WH_2 satisfying Theorem 2.

一方，在庫数が 35 の場合では，WT_1 の注文 $\mathcal{T}_1^1$ の実行時に在庫不足ということで取寄せによる入荷 $\mathcal{T}_2^1$ が実行されることになる．$\mathcal{T}_2^1$ による入荷日が WT_4 の配達 $\mathcal{T}_4^4$ の配達日より先なら，**図 6** に示された WH_3 のように WT_1 の配達 $\mathcal{T}_4^1$ を先に行う並行実行が可能となる．WH_3 は，出力データの隔離性と内部入力データの隔離性を満たすが，定理 2 は満たさない．これは，WT_1 の配達 $\mathcal{T}_4^1$ から入荷 $\mathcal{T}_2^1$ までの間，結果条件 $x_3 \geq 30$（在庫数は注文数以上）が満たされないためである．したがって，定理 2 は出力データの隔離性と内部入力データの隔離性を満たすための十分条件を与えている．

タスクの入力条件は与えられており，トランザクションの結果条件はその定義から求められる．内部入力データの隔離性と出力データの隔離性の必要十分条件をそれぞれ，

(1) トランザクション内の各プロセスが実行されるときに入力条件を満たす．
(2) トランザクションの終了時に，データベースにおけるトランザクションの使用データ項目の値は結果条件を満たす．

とすることができる．

条件 (1) を満たせば，各プロセスの出力データは出力条件を満たす．したがって，トランザクションの使用データ項目がトランザクションの終了まで他のトランザクションによって変更されなければ，条件 (2) を満たす．他のトランザクションに変更されてもその変

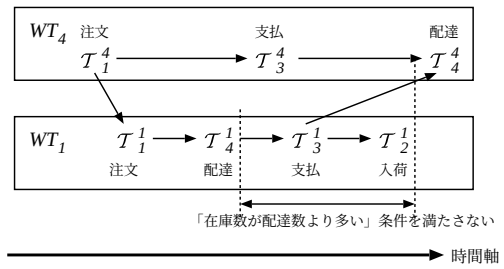

図 6 隔離性は満たすが定理 2 は満たさない WH_3
Fig. 6 WH_3 satisfying isolation but not Theorem 2.

更が各時点での結果条件を損なわない範囲であればよい．WH_2 はその例である．さらに，その変更がある時点での結果条件を損なっても条件 (1) と (2) を満たせばよい．WH_3 はその例である．定理 2 に基づく制御法は，つねに出力データの隔離性と内部入力データの隔離性を保証する安全な方法であるが，条件 (1) と (2) に基づく制御法の場合は，プロセスの入力条件が満たされておらず，その実行が待たされることがある．これは楽観的な実現法に相当する．

5. 外部入力データの隔離性

外部入力データの隔離性は，各タスクの実行が他のトランザクションの一貫していない中間結果を検索しないことを意味する．データベースの一貫性に関する情報がない場合には，各トランザクションが一貫したデータベース上の遷移であるという仮定で一貫したデータベースを識別することになる．すなわち，各タスクの実行が他のトランザクションの中間結果を検索していなければ，外部入力データの隔離性は満たされる．本章ではこの考え方の下での判定方法を導入する．

定義 5 トランザクション $WT_i(TN_i,TE_i)$ $(i=1,2,\cdots)$ からなるスケジュール $WH(HN,HE)$ に対し，次のグラフ $RI(RN,RE)$ を WH の**外部入力データの隔離性判定グラフ**という．

- 節点集合 RN：プロセス $\mathcal{T} \in WT \in WH$ とトランザクション $WT \in WH$ から構成される．
- 枝集合 RE：プロセス $\mathcal{T}$ の入力データがトランザクション WT によって変更されれば，$\mathcal{T}$ から WT への rw 枝が存在する．また，WT の出力データ，または $\mathcal{T}_j$ の出力データが $\mathcal{T}$ の入力データとして検索されれば，WT から $\mathcal{T}$ へ，または $\mathcal{T}_j$ から $\mathcal{T}$ への wr 枝が存在する． □

例 6 ホテルの予約 t_a とキャンセル t_b，切符の予約 t_c とキャンセル t_d，支払 t_e の 5 つのタスクからなるワークフロー CF_3 を考える（**図 7**）．
その一部のタスクの定義を次のようなものとする．

- ホテル予約 t_a：

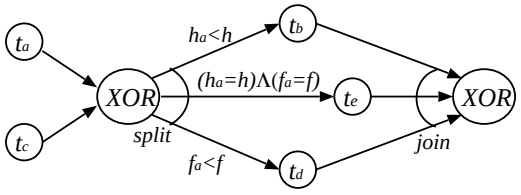

図 7 制御フロー CF_3
Fig. 7 Control Flow CF_3.

I_a={ 利用可能数:$count_h$, 予約数:h},
O_a={ 利用可能数:$count_h$, 回答数:h_a},
$IC_a = \{count_h^I \geq 0, h > 0\}$,
$OC_a = \{count_h^O = count_h^I - h_a, count_h^O \geq 0\}$.

- ホテルキャンセル t_b：
I_b={ 利用可能数:$count_h$, キャンセル数:c_h},
O_b={ 利用可能数:$count_h$},
$IC_b = \{c_h > 0\}$,
$OC_b = \{count_h^O = count_h^I + c_h\}$.

- 切符予約 t_c：
I_c={ 利用可能数:$count_t$, 予約数:f},
O_c={ 利用可能数:$count_t$, 回答数:f_a},
$IC_c = \{count_t^I \geq 0, f > 0\}$,
$OC_c = \{count_t^O = count_t^I - f_a, count_t^O \geq 0\}$.

図 8 は，ホテルを予約して切符を予約をし，両方が予約できたのち支払を行うトランザクション WT_5 と，切符を予約してホテルを予約をし，両方が予約できたのち支払を行うトランザクション WT_6 からなるスケジュール WH_4 を示している．ホテル予約から見れば実行順序は $\mathcal{T}_a^5$ から $\mathcal{T}_a^6$ の順であり，切符支払から見れば実行順序は $\mathcal{T}_c^6$ から $\mathcal{T}_c^5$ の順である．したがって，WH_4 の直列可能性判定グラフ[4]には WT_5 と WT_6 間に巡回が存在し，WH_4 は直列可能ではない．
WH_4 の外部入力データの隔離性判定グラフ $RI_4(RN_4, RE_4)$ を**図 9** に示している．$\mathcal{T}_c^5$ と $\mathcal{T}_a^6$ は互いに相手のトランザクションの結果しか検索していないので，$RI_4(RN_4, RE_4)$ は非巡回である． □

あるタスク t の実行 $\mathcal{T}$ が WT_j の中間結果を検索したとする．$\mathcal{T}$ が WT_j の結果を検索したので，WT_j と $\mathcal{T}$ 間に write-read 依存性がある．外部データの隔

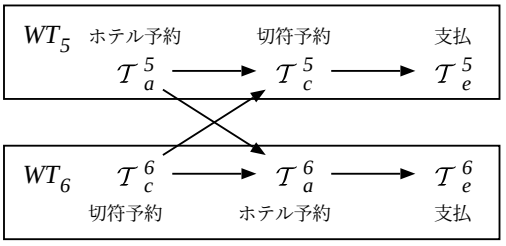

図 8 相互参照スケジュール WH_4
Fig. 8 Mutual reference schedule WH_4.

$$WT_6 \overset{w}{\rightarrow} \overset{r}{\mathcal{T}_c^5}$$
$$WT_5 \overset{w}{\rightarrow} \overset{r}{\mathcal{T}_a^6}$$

図 9 WH_4 の外部入力データの隔離性判定グラフ
Fig. 9 Outer input data isolation graph $RI_4(RN_4, RE_4)$.

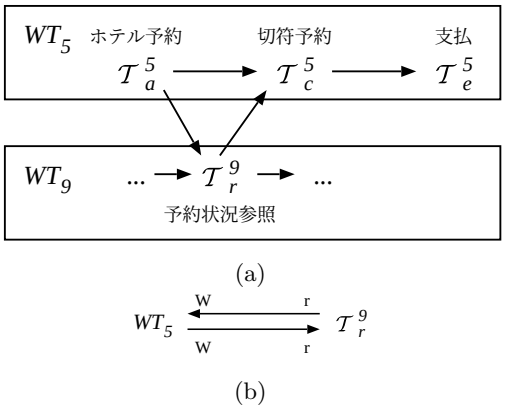

図 10 中間結果参照スケジュール WH_6
Fig. 10 WH_6 with intermediate result reference.

離性判定グラフ $RI(RN, RE)$ においては，WT_j から $\mathcal{T}$ への wr 枝となる．また，$\mathcal{T}$ の入力データ項目が WT_j によって変更されるので，$\mathcal{T}$ と WT_j 間は read-write 依存性がある．$RI(RN, RE)$ においては，$\mathcal{T}$ から WT_j への rw 枝となる．したがって，判定グラフに $\mathcal{T}$ と WT_j 間の wr 枝と rw 枝からなる巡回が存在することになる．

例 7 切符の予約を行うトランザクション WT_5 と予約状況を参照するタスク t_r のプロセス $\mathcal{T}_r^9$ を含むトランザクション WT_9 が並行に実行されたとする．$t_r(I_r, O_r, IC_r, OC_r)$ は $I_r = \{h_a, f_a\}$，$O_r = IC_r = OC_r = \phi$ である．そのスケジュール WH_6 が**図 10** (a) であったとすると，WH_6 の外部入力データの隔離性判定グラフは図 10 (b) になる．$\mathcal{T}_r^9$ と WT_5 の間に wr 枝と rw 枝からなる巡回が存在し，隔離性を満たさない． □

一般の場合に拡張するため，中間結果を検索していないことを示す非巡回性を定義する．

定義 6 トランザクション $WT_i(TN_i, TE_i)$ $(i = 1, 2, \cdots)$ からなるスケジュール $WH(HN, HE)$ の外部入力データの隔離性判定グラフ $RI(RN, RE)$ において，各 $\mathcal{T} \in TN_i$ に対して WT_j から $\mathcal{T}$ への推移的な wr 枝と $\mathcal{T}$ から WT_j への rw 枝からなる閉路が存在しないとき，$RI(RN, RE)$ は**中間値参照非巡回**という． □

定理 3 トランザクション $WT_i(TN_i, TE_i)$ $(i = 1, 2, \cdots)$ からなるスケジュール $WH(HN, HE)$ の外部入力データの隔離性判定グラフ $RI(RN, RE)$ が中間値参照非巡回なら，各 $\mathcal{T} \in TN_i$ は他のトランザクションの中間結果を検索しておらず，$WH(HN, HE)$ は外部入力データの隔離性を満たす．

証明． $RI(RN, RE)$ は中間値参照非巡回であるが，ある $\mathcal{T}_s \in TN_i$ は他のトランザクション WT_j の中間結果を検索していたとする．

$\mathcal{T}_s$ が直接または間接に参照した出力データを作成したトランザクションは，$RI(RN, RE)$ において $\mathcal{T}_s$ への wr 枝からなる経路が存在する．したがって，WT_j から $\mathcal{T}_s$ への wr 枝からなる経路が存在する．一方，$\mathcal{T}_s$ の入力データを変更したトランザクションは，$\mathcal{T}_s$ からの rw 枝が存在する．したがって，$\mathcal{T}_s$ から WT_j への rw 枝が存在する．このため，$\mathcal{T}_s$ と WT_j の間に閉路が存在し，$RI(RN, RE)$ は中間値参照非巡回ではない． □

WH_3 の外部入力データの隔離性判定グラフも非巡回である．これはどのプロセスも 2 つ以上の共有データ項目を参照していないためである．したがって，WH_3 は入出力データの隔離性を満たす．一方，WH_4 において，WT_5 と WT_6 のホテルや切符の利用可能数が 0 以上という条件は，関連する予約プロセスの実行終了からトランザクションの終了まで保持されている．すなわち，定理 2 を満たし，WT_5 と WT_6 は出力データと内部入力データの隔離性を満たす．したがって，WH_4 も入出力データの隔離性を満たす．

6. 議　　論

ワークフローにおける一貫性情報には，データベースが一貫するための必要条件（または必要十分条件）となる一貫性制約と，タスクの入出力条件で表されるアプリケーションの一貫性情報の 2 種類がある．本論文では，後者のみを利用した場合について検討を行い，タスクの入力データの一貫性を保証する入力データの隔離性と，実行結果のデータベースの一貫性を保証する出力データの隔離性を提案した．論文 3) では，一貫性制約で表されるデータベースの一貫性の必要十分条件を利用できると仮定しているため，各プロセスの出力条件の保持期間はデータフローの子節点となっている．本論文では，データベースの一貫性の必要十分条件が定義できない場合は，出力条件を保持すべき期間は，トランザクションの終了までであることを示した．出力一貫性はトランザクションを単位とするので，出力データの隔離性もトランザクション単位である必要がある．

外部入力データの隔離性については，競合の種類を区別し，節点の単位を細分することにより直列可能性の拡張となっている．従来の直列可能性判定グラフでは，トランザクションを節点としており，2 つのトラン

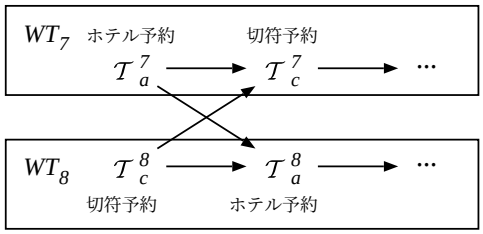

図 11 スケジュール WH_5
Fig. 11 Schedule WH_5.

ザクション間に競合する操作があれば枝が存在する[4]．しかし，直列可能性判定グラフの閉路中の競合関係に rw の競合が存在しなければ，外部入力データの隔離性を満たす．これは中間値参照非巡回となるためである．

出力データの隔離性と内部入力データの隔離性については，タスクの入出力条件であるアプリケーションの一貫性情報を用いて必要十分条件や十分条件を与えた．一貫性情報や述語に基づく競合操作の実行順序の変更に関する研究は，以前から行われている[2),5)]．ワークフローにおいてタスクが原子性の単位であるため，各時点で満たさなければならない条件を結果条件として正確に定義できる．それによって，スケジュール全体として操作単位の実行順序の変更を考えることができるようになった．たとえば，Escrow アルゴリズム[5)]では WH_3 といったスケジュールは実行不可能であるが，本論文の正当性基準では実行可能である．

本論文で提案している入出力データの隔離性によって許されるスケジュールには，直列実行においては生成されることのない実行を含む．たとえば，例 6 において，それぞれホテルの予約，切符の予約と切符の予約，ホテルの予約の順で実行される WT_7 と WT_8 が **図 11** のように実行され，予約数はすべて 2 とする．WH_5 が実行されるときにホテルと切符の利用可能数もすべて 2 とすると，WT_7 と WT_8 の両方ともホテルか切符かの一方しか予約が取れない．予約をキャンセルするタスク，あるいはキャンセル待ちのタスクがあれば，そのようなタスクが実行され，実際の予約業務に近いスケジュールを可能とする．直列可能性に基づく制御ではこのような実行は許されず，後退復帰などにより一方の予約が完了するスケジュールのみが生成される．

7. おわりに

本論文では，ワークフローに従う業務が並行に実行される場合を想定して，ワークフロートランザクションの隔離性について検討した．隔離性は，単独実行時に一貫したデータベース上の遷移であるトランザクションからなる並行実行を問題なく行うために必要なものである．このため，トランザクションの一貫性に対応して隔離性の見直しを行った．ワークフロートランザクションの一貫性が入力データと出力データに分けて定義されるため，隔離性もそれに応じた定義を導入した．出力データの隔離性と内部入力データの隔離性については，タスクの入出力条件を用いている．外部入力データの隔離性についても，互いに相手の最終結果を検索している直列可能でないスケジュールを含む．また，WH_5 のような直列可能実行で生成されることのない実際の作業に近いスケジュールも実行できる．

本論文で示したワークフロートランザクションの隔離性を用いれば，ワークフローの特徴を生かした並行処理制御方式が可能となる．そのような制御方式について安全な制御法から楽観的な制御法まで幅広く比較検討していく予定である．

参考文献

1) Alonso, G., Agrawal, D., Abbadi, A.E. and Mohan, C.: Functionality and Limitations of Current Workflow Management Systems, *IEEE Expert: Special Issue on Cooperative Information Systems* (1997).
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) Arpinar, L.B., Halici, J., Arpinar, S. and Dogac, A.: Formalization of Workflows and Correctness Issues in the Presence of Concurrency, *Distributed and Parallel Databases*, Vol.7, No.2, pp.199–248 (1999).
4) Bernstein, P.A., Hadzilacos, V. and Goodman, N.: *Concurrency Control and Recovery in Database Systems*, Addison-Wesley (1987).
5) Gray, J. and Reuter, A.: *Transaction Processing: Concepts and Techniques*, Morgan Kaufmann (1994).
6) Leymann, F. and Roller, D.: Business Process Management with FlowMark, *Proc. IEEE Compcon* (1994).
7) Kamath, M. and Ramamritham, M.: Correctness Issues in Workflow Management, *Distributed Systems Engineering Journal, Special Issue on Workflow Management Systems* (1997).
8) Puustjarvi, J., Tirri, H. and Veijalainen, J.:

Concurrency Control for Overlapping and Cooperative Workflows, *IEEE Trans. Computer Systems Bulletin*, pp.24–30 (1996).
9) Puustjarvi, J.: Workflow Concurrency Control, *Computer Journal*, Issue 1, pp.42–53 (2001).
10) Ramamritham, K. and Chrysanthis, P.K. (Eds.): *Advances in Concurrency Control and Transaction Processing*, IEEE Computer Society Press (1997).
11) Reuter, A. and Schwenkreis, F.: ConTracts — A Low-Level Mechanism for Building General-Purpose, *Workflow Management Systems, IEEE Bulletin of the Technical Committee on Data Engineering*, Vol.18, pp.4–10 (1995).
12) Rusinkiewicz, M. and Sheth, A.P.: Specification and Execution of Transactional Workflows, *Modern Database Systems: The Object Model, Interoperability, and Beyond*, pp.592–620 (1995).

(平成 14 年 10 月 7 日受付)
(平成 15 年 1 月 15 日採録)

（担当編集委員　清木 康，市川 哲彦，佐藤 聡，
原 隆浩，細川 宜秀）

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．同大学助教授を経て，平成 15 年度より同大学情報工学部情報工学科教授．平成 12 年度米国カリフォルニア大学サンタバーバラ校客員研究員．並行処理制御，WEB 型教育支援システム等の研究に従事．ACM，IEEE，電子情報通信学会，日本ソフトウェア科学会各会員．

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，大型計算機センター講師，同助教授，経済学部助教授を経て，現在同大学大学院経済学研究院助教授．この間，平成 7 年米国パデュ大学客員研究員．データベースの設計論・質問処理論，情報システムの研究に従事．電子情報通信学会，ACM，IEEE，日本 OR 学会等会員．