C1-1 トランザクショナルワークフローにおける並行実行の正当性

徐　海燕[†]　古川哲也[††]　史　一華[†††]

ワークフロー管理システムが今日の情報社会に果たす役割が益々増えている。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が検索・更新されるデータ項目が重なっていれば、並行処理制御が必要となる。そのためには、ワークフローの特徴に対処できなければならない。本論文では、ワークフローに従う作業が並行に実行されるときの正当性基準について検討する。まず、反復処理も表現できるワークフローモデルを定義する。ワークフロー内の作業の流れを表す制御フローに対して、表現式による記述方法を導入する。それによって、ワークフローに従う実行、すなわち、トランザクションは簡単に記述できる。次に単独実行されるトランザクション中のデータが満たされる性質を分析し、並行実行の正当性基準を提案する。

keywords: ワークフロー、並行実行、並列実行、正当性基準、トランザクション、一貫性制約

1. はじめに

近年、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。ワークフロー管理システムが対象とする業務には定型か非定型かという2種類のものがある。現在、稼働中のワークフロー管理システムのほとんどは、定型業務を対象にしたものである。定型業務の代表的なものとして、資料購入、与信審査などが挙げられる。この種の業務は、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。本論文では、定型的な処理を行うワークフロー管理システムに焦点を当てて検討を行う。

ワークフロー内での作業の基本単位は、アクティビティ、すなわちタスクである。ワークフローが実行されると、その中のタスクは制御フローに従って実行される。各タスクは先行のタスクやデータベースからデータを入力し、処理を行い、結果を後続のタスクまたはデータベースへ出力する。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が操作するデータが重なることがある。すなわち、ワークフロー処理に共有データが存在する。共有データが存在していれば、並行処理制御が必要となる。以下はその一例である [10]。

例 1 クレジットカードの申請を処理する業務では、クライアントの信頼性を審査することによって、許可する額を決める。すなわち、システムにはクライアントの財産、収入、債務より、許可できる使用額を制限する一貫性制約が存在する。しかし、並行実行の制御を行わないと、同一クライアントが同時に2枚のクレジットカードを申請すれば、両方とも許可されることになる。すなわち、使用できる総額が許可できる額の倍になってしまうという一貫性制約を満たさない事態を招く。

ワークフローの並行実行に対して制御の必要性があり、そのため、トランザクショナルワークフローという概念が導入されている [8]。ワークフローの基本単位は入出力データに対する条件を持つタスクであり、ワークフロー管理システム全体にもデータに対して一貫性制約がある。このため、ワークフロー管理システムに適した並行実行の正当性基準と制御方式が必要である。

ワークフローとデータベースの両方の視点から、この課題に対する研究が行われている [3),7]。しかし、問題の全面的な解決には至っていない。本論文では、ワークフローの形式的な記述方法を導入することによって、単独実行と同じ性質を保証できる並行実行の正当性基準を提案する。

本論文では、次のような仮定に基づいて議論を行う。

- ワークフロー管理システムのデータは、データベースに記憶される。
- データベースの一貫性は、データが一貫性制約を満たしているかどうかで判断する。

ワークフローの処理の流れを表す制御フローに対しては、細分されたそれぞれの部分に対する議論を行いやすくするため、本論文では、表現式による記述方法を導入する。それによって、制御フロー内の反復業務は、対応する表現式 r の反復 r^+ で表現できる。r の

† 福岡工業大学情報工学部情報工学科
†† 九州大学大学院経済学研究院
††† 西南学院大学商学部

表す集合を R とすると、R^+ は r^+ の表す集合となる。したがって、巡回制御フローに対しても、トランザクションを「対応する表現式の表す集合の要素」というように簡潔に定義できる。次に本論文では、各トランザクションが単独に実行されるときに、その中の各タスクの入出力データが満たす性質について考察する。それと同じような性質を満たす方針の下で並行実行の正当性基準を提案する。正当な並行実行は、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ことを示す。

本論文は、次のように構成される。2 章では、制御フローとそれに対応する表現式を形式的に定義し、ワークフローを導入する。3 章では、ワークフローの実行可能性について考察する。4 章では、トランザクションを定義し、ワークフローの妥当性と単独実行時の各タスクの性質を分析する。単独実行時と同じ性質を満たす方針の下で、5 章では、並行実行の正当性基準を導入し、その特徴について検討する。関連研究との比較は 6 章で行い、7 章は全体のまとめである。

2. ワークフロー

本章では、ワークフローの形式的な定義を与える。ワークフローは、原子性の単位であるタスク、タスク間の実行順序を与える制御フロー、データベースの一貫性制約によって記述される。

2.1 タ ス ク

ワークフローの基本的な単位は、アクティビティ、すなわちタスクである。各タスクはその特徴を表す 5 つのパラメータによって記述される [6)]。

定義 1 タスク t は, 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、入力データ以外に検索されるデータ項目の集合 RS からなる組 $t = (I, O, RS, IC, OC)$ によって記述される。

I は、タスクの処理を特定するデータ項目の集合である。RS は、タスクの処理を実行するために必要なデータ項目の集合である。

並行実行において、タスク t は原子性の単位である。入力データ I が入力条件 IC を満たすならば、タスク t によって生成される出力データ O は、出力条件 OC を満たすと仮定する。また、本論文では、$\epsilon = (\phi, \phi, \phi, \phi, \phi)$ で何もしない特別なタスクを表す。

例 2 例 1の業務におけるタスクは、次のように形式的に記述できる。

- 受付 t_0:
  I_0 ={ クライアント ID x_1}, $RS_0 = \phi$, O_0 ={ 受付番号 x_3}, $IC_0 = \phi$, OC_0 ={ クライアント x_1 には受付番号 x_3 がある }。
- 予備審査 t_1:
  I_1 ={ クライアント ID x_1}, $RS_1 = \{x_1$ の所有のクレジットカードの数 $x_4\}$, $O_1 = \phi$, $IC_1 = \{x_4 = 0\}$、$OC_1 = \phi$。
- 審査 t_2:
  I_2 ={ クライアント ID x_1}, RS_2 ={ 申請額 x_2、財産 x_5、収入 x_6、債務 x_7}, $IC_2 = \phi$、O_2 = { 許可額 x_8、カード番号 x_9、所有カードの数 x_4}, OC_2 ={ 財産、収入と債務で使用上限を決める規則 }。

すなわち、t_0 は、クライアントの申請額をデータベースに記憶し、受付番号を配布する。t_1 は、申請者がクレジットカードをすでに所有していないことを検査する。t_2 は、t_1 の検査に合格できた申請者に対して、許可額を審議する。入力データ I と RS の違いは、例えば、予備審査 t_1 を行うためにはクライアント ID x_1 が必要なので、それは入力データとなる。t_1 の実行では、ID x_1 を用いてクレジットカードの所有状況 x_4 が検索される。

クレジットカードを 2 枚以上持つことができないのであれば、データベースの一貫性制約は次のようになる。

$C_1 : x_4 \leq 1 \wedge OC_2$

2.2 制御フローとその表現式

タスクの実行順序を、タスクを節点とする有向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、並列 (AND-分離)、撰択 (OR-分離)、条件分岐 (XOR-分離) がある。有向グラフにより表現を用いると、実行の流れを理解しやすいが、一方で並行実行時の性質を検討するためには、制御フローを細分して議論する必要がある。そのためには、制御フローに対応する表現式を用いた方が有効であり、本論文では、両方の記述方法を用いる。

定義 2 制御フローとそれに対応する表現式は、次のように再帰的に構築された有向グラフ $CF = (N, E, q, f)$ とタスク集合 Σ 上の順次、反復、AND-分離、XOR-分離の 4 つの演算からなる式 r である。ここで、N は節点集合、E は枝集合、q は始点、f は終点である。

ϵ と Σ 中の t に対して、それぞれの制御フローは $CF = (\{\epsilon\}, \{\}, \epsilon, \epsilon)$、$CF = (\{t\}, \{\}, t, t)$ である。対応する表現式は、それぞれ ϵ と t で、演算の個数が 0 である。

対応する表現式が r_j である制御フロー $CF_j = (N_j, E_j, q_j, f_j)(j = 1, 2, \cdots, k)$ の各実行順序を、次のような制御フロー CF と、演算の個数が 1 以上であ

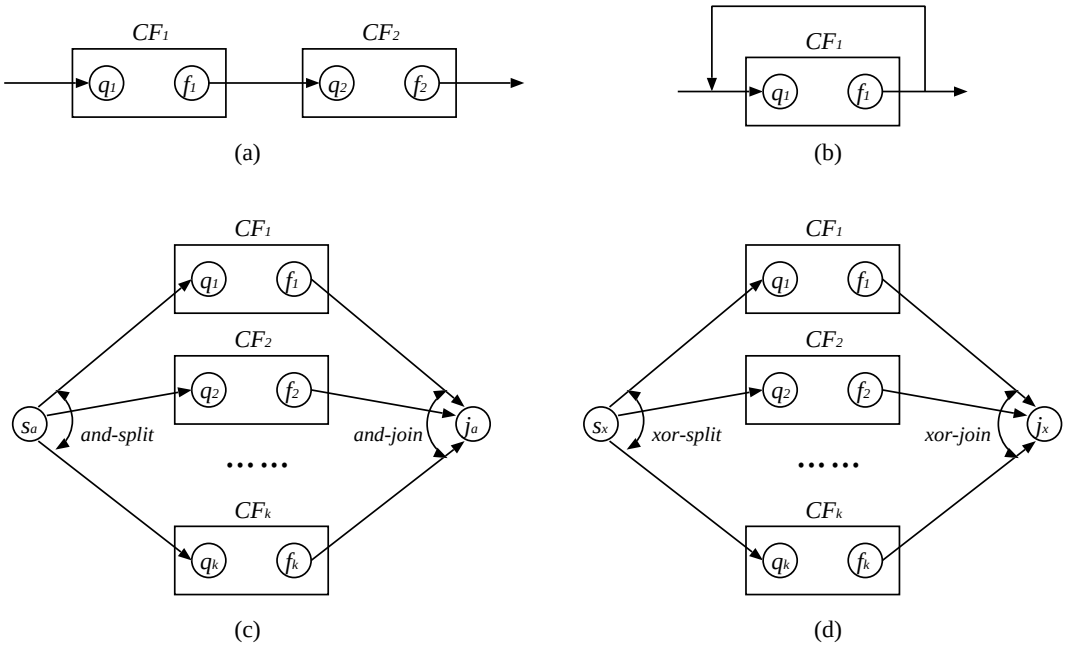

図 1　定義 2 の帰納法のステップの構成 (a) 順次。(b) 反復。(c)AND-分離。(d)XOR-分離

る対応する表現式 r で記述する。

- CF_1 と CF_2 の順次実行。CF は、CF_1 の終点と CF_2 始点をつなぐ、すなわち、$CF = (N_1 \cup N_2, E_1 \cup E_2 \cup \{(f_1, q_2)\}, q_1, f_2\})$ である (図 1(a))。
  対応する表現式 r は、$r = r_1 \cdot r_2$ である。
- CF_1 の反復実行。CF は、CF_1 の終点と始点をつなぐ、すなわち、$CF = (N_1, E_1 \cup \{(f_1, q_1)\}, q_1, f_1)$ である (図 1(b))。
  対応する表現式は、$r = r_1^+$ である。
- CF_1,CF_2,$\cdots$,CF_k の並列実行。CF は、CF_i の始点とつなぐ AND-分離節点 s_a と、CF_i の終点とつなぐ AND-結合節点 j_a を設け、それらをそれぞれ始点、終点とする、すなわち、$CF = (\bigcup N_i \cup \{s_a, j_a\}, \bigcup E_i \cup \{(s_a, q_i), (f_i, j_a) \mid i = 1, 2, \cdots, k\}, s_a, j_a)$ である (図 1(c))。
  対応する表現式 r は、$r = (r_1, r_2, \cdots, r_k)$ である。
- CF_1,CF_2,$\cdots$,CF_k の撰択実行。CF は、CF_i の始点とつなぐ XOR-分離節点 s_x と、CF_i の終点とつなぐ XOR-結合節点 j_x を設け、それらをそれぞれ始点、終点とする、すなわち、$CF = (\bigcup N_i \cup \{s_x, j_x\}, \bigcup E_i \cup \{(s_x, q_i), (f_i, j_x) \mid i = 1, 2, \cdots, k\}, s_x, j_x)$ である (図 1(d))。
  対応する表現式 r は、$r = r_1 \bigoplus r_2 \bigoplus \cdots \bigoplus r_k$ である。

反復実行を表すため、CF は有向巡回グラフである。分離節点と結合節点は一対一に対応する。Σ をタスクの集合とすると、CF の節点集合 N は、Σ と分離/結合節点の集合の和集合となる。一般によく使われている OR-分離は、XOR-分離と AND-分離によって表せるため、省略する。

タスク集合 Σ 上の表現式が表すタスクの実行系列の集合を、次のように再帰的に定義する。

(1)　ϵ の表す集合は $\{\epsilon\}$ である。

(2)　Σ 中の各 t の表す集合は $\{t\}$ である。

(3)　r と s がそれぞれ集合 R と S を表す表現式のとき、順次演算 $r \cdot s$ と反復演算 r^+ は、それぞれ集合 $RS = \{xy \mid x \in R, y \in S\}$ と R の正閉包 R^+ を表す。

(4)　r_1, r_2, $\cdots$, r_k をそれぞれ集合 R_1, R_2, $\cdots$, R_k を表す表現式とする。AND-分離演算 $(r_1, r_2, \cdots, r_k)$ の表す集合は $R_j (1 \leq j \leq k)$ の直積 $R_1 \times R_2 \times \cdots \times R_k$ である。

(5)　r_1, r_2, $\cdots$, r_k をそれぞれ集合 R_1, R_2, $\cdots$, R_k を表す表現式とする。XOR-分離演算 $(r_1 \bigoplus r_2 \bigoplus \cdots \bigoplus r_k)$ の表す集合は $R_j (1 \leq j \leq k)$ の和集合 $\bigcup_{j=1}^{k} R_j$ である。

ここでは、r^+ のみ定義しているが、r^* は $(r \bigoplus \epsilon)^+$ によって表せる。

例えば、$(s_1, s_2) \cdot (s_1, s_2) \cdot (s_1, s_2) \cdot s_3$ は、$(s_1, s_2)^+ \cdot s_3$ で表される集合の要素であるが、$(s_1 \bigoplus s_2)^+ \cdot s_3$ で表される集合の要素ではない。一方、$s_1 \cdot s_2 \cdot s_2 \cdot s_2 \cdot s_1 \cdot s_3$ は、$(s_1 \bigoplus s_2)^+ \cdot s_3$ で表される集合の要素である。

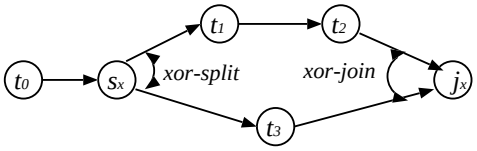

図 2　制御フロー $CF1$

例 3　例 1の業務に対応する制御フロー CF_1 を図 2に示している。t_3 は、申請者がすでにカードを所有している場合の例外処理を行うタスクである。

$I_3 = I_1,\ O_3 = \phi,\ RS_3 = RS_1,\ IC_3 = \neg IC_1,\ OC_3 = \phi$。

CF_1 の表現式 r_1 は、次のようになる。

$r_1 = t_0 \cdot (t_3 \bigoplus (t_1 \cdot t_2))$

制御フローと一貫性制約を用いてワークフローを定義する。

定義 3　ワークフロー WF は、タスク集合 Σ、制御フロー CF または対応する表現式 r、一貫性制約 C からなる組 $WF = (\Sigma, CF, C)$ または $WF = (\Sigma, r, C)$ である。

例 4　例 1のクレジットカード審査ワークフロー WF_1 は、$\Sigma_1 = \{\epsilon, t_0, t_1, t_2, t_3\}$ に対して、$WF_1 = (\Sigma_1, CF_1, C_1)$ または $WF_1 = (\Sigma_1, r_1, C_1)$ となる。

ワークフロー WF のデータフローについては、本論文では特に制限しない。同一のワークフロー WF に対して、タスクの出力データをデータベースを系由して後続のタスクに渡すかどうかによって、多くの種類のデータフローが定義できる。本論文では、データフローに依存しない正当性基準を目指す。すなわち、各タスクの出力データをすべてデータベースを系由して後続のタスクに渡す場合においても適用できる正当性基準について検討する。

一貫性制約 C、入力条件 IC や出力条件 OC は、第一階述語論理の節

$$B_1, \cdots, B_m \leftarrow A_1, \cdots, A_n$$

の形で定義される。ただし，A_i,B_j に現れるのはデータベースの項目またはデータベース上の全称束縛変数である。節 $B_1, \cdots, B_m \leftarrow A_1, \cdots, A_n \equiv \neg(A_1 \wedge \cdots \wedge A_n) \vee (B_1 \vee \cdots \vee B_m) \equiv \neg A_1 \vee \cdots \vee \neg A_n \vee B_1 \vee \cdots \vee B_m$ であるため、1 階述語論理の表現力を持つ和積標準形は、節集合でも表現できる。述語 (A_i や B_j) に関わるデータ項目とは、述語に現れる定数または変数のドメインに属するものである。節中の述語に関わるデータ項目を、節に関わるデータ項目という。データ項目集合 X の値が制約 C(IC または OC) を満たすとは、X にその値を代入すると、X 以外のデータ項目の値にかかわらず全ての節の値が真となることをいう。

3. ワークフローの実行可能性

本章では、ワークフロー WF に対して、その入力データ、入力条件を計算する方法を定義し、その性質を分析する。

定義 4　制御フロー CF に対応する表現式 r に対して、次のように再帰的にパラメータ I, RS, IC を定義する。

演算の個数が 0 である表現式 r は、Σ 中の要素 t と ϵ である。t の I, RS, IC と ϕ, ϕ, ϕ がそれぞれのパラメータとなる。

表現式 $r_j (j = 1, 2, \cdots, k)$ のパラメータを I_j,RS_j,IC_j を r_j とすると、それらを演算で用いる表現式 r のパラメータ I, RS, IC は、次のようにする。

- 順次 $r = r_1 \cdot r_2$。
  $I = I_1 \cup (I_2 - O_1)$
  $RS = RS_1 \cup RS_2$
  $IC = IC_1 \wedge$ (IC_2 中の $I_2 - O_1$ のみに関わる節)
- 反復 $r = r_1^+$。
  I, RS, IC は、I_1, RS_1, IC_1 と同じである。
- AND-分離 $r = (r_1, r_2, \cdots, r_k)$。
  $I = \bigcup_{i=1}^{k} I_i$
  $RS = \bigcup_{i=1}^{k} RS_i$
  $IC = \wedge_{i=1}^{k} IC_i$
- XOR-分離 $r = r_1 \bigoplus r_2 \bigoplus \cdots \bigoplus r_k$。
  $I = \bigcup_{i=1}^{k} I_i$
  $RS = \bigcup_{i=1}^{k} RS_i$
  $IC = \vee_{i=1}^{k} IC_i$

表現式のパラメータは、実行順序にかかわらず一意に決められる。

ワークフローの定義が不十分であれば、タスクがその入力条件を満たさないために実行できなくなる場合が生じる。一貫したデータベースに対して全ての業務が実行できるように定義されている必要がある。そのため、ワークフローの実行可能性を定義する。

定義 5　$WF = (\Sigma, r, C)$ は、表現式 r の一貫性制約 C を満たす入力データ I に対し、次のような性質を満たすならば、実行可能なワークフローである。

(1)　I は、入力条件 IC を満たす。

(2)　WF 中の各タスク t_i の入力データ I_i は、t_i の入力条件 IC_i を満たす。

入力条件を満たさない入力データに対しては、例外処理を設ければよい。例えば、条件 (1) を満たさない $r = (I, O, RS, IC, OC)$ に対しては、例外処理 $r' = (I, \phi, RS, \neg IC, \phi)$ を、$r' \bigoplus r$ というように設けることになる。このため、妥当でない $WF = (\Sigma, r, C)$

は、妥当な WF に定義し直せる。

4. トランザクショナルワークフロー

本章では、ワークフローに従う実行であるトランザクションを導入し、それを用いてワークフローの妥当性について検討する。

4.1 トランザクション

本節では、トランザクションを定義し、そのグラフ表現を与える。

定義 6 $WF = (\Sigma, r, C)$ の r で表される集合 R の要素のインスタンスを、WF の**トランザクション** WT という。

r で表される集合 R の要素を、トランザクションの表現式という。トランザクションの表現式に対しても、定義 2で与えられた方法でそのグラフ表現を構築できる。ただし、トランザクションの表現式には反復演算 $^+$ と XOR 分離演算 $\bigoplus$ を含まない。トランザクションのグラフ表現 $WT(TN, TE)$ は、その表現式のグラフ表現の節点をインスタンス化したものである。すなわち、$WT(TN, TE)$ は、次のような有向非巡回グラフである。節点集合 TN は、重複を許したタスクのインスタンス集合、AND-分離節点集合、AND-結合節点集合からなる。AND-分離節点と AND-結合節点は、一対一に対応する。

例 5 例 1において、ある申請が審査に合格したときのトランザクション WT_1 のグラフ表現は、図 3のようになる。ここで、α_i を $t_i (i = 0, 1, 2)$ のインスタンスとする。

$$\alpha_0^1 \rightarrow \alpha_1^1 \rightarrow \alpha_2^1$$

図 3 トランザクション WT_1

r の制御フロー $CF = (N, E, q, f)$ と $WF = (\Sigma, r, C)$ のトランザクションのグラフ表現の対応を見てみる。制御フロー $CF = (N, E, q, f)$ が非巡回の場合、$WF = (\Sigma, CF, C)$ のトランザクションのグラフ表現は、CF の始点から終点への経路のインスタンスとなる。巡回の場合は、トランザクションのグラフ表現は、各閉路について繰り返しの回数が特定し、展開したグラフのインスタンスである。

4.2 トランザクショナルワークフローの性質

トランザクションレベルでの出力データと出力データの性質を用いてワークフローの妥当性を定義する。

定義 7 トランザクションの表現式 r に対して、次のように再帰的にパラメータ O, OC を定義する。

演算の個数が 0 である表現式 r は、Σ 中の要素か ϵ である。それぞれのパラメータが r のパラメータになる。

演算の個数 $i \geq 1$ である r は、次のように i より少ない $r_j (j = 1, 2, \cdots, k)$ に分解でき、パラメータ O, OC を計算する。

- 順次 $r = r_1 \cdot r_2$。
  $O = O_2 \cup (O_1 - O_2)$
  $OC = OC_2 \wedge$ (OC_1 中の $O_1 - O_2$ のみに関わる条件)
- AND-分離 $r = (r_1, r_2, \cdots, r_k)$。
  $O = \bigcup_{i=1}^{k} O_i$
  $OC = \wedge_{i=1}^{k} OC_i$

タスクの出力データには、図 4で示されるように後続タスクに渡される中間データとデータベースへの最終結果がある。ワークフローは、そのような出力データの性質を考慮してデータベースの一貫性を保持しなければならない。

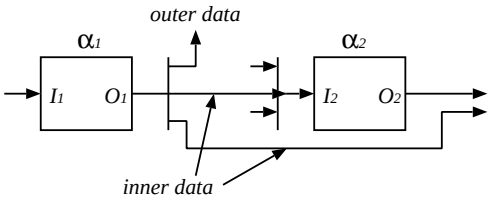

図 4 出力データの種類

定義 8 実行可能な $WF = (\Sigma, r, C)$ 中の表現式 r と WF の各トランザクションが、次のような性質を満たす時、WF は妥当なワークフローである。

(1) 各トランザクションは、入力条件 IC、出力条件 OC と一貫したデータベースで実行されることより実行結果のデータベース性を保証できる。

(2) WF 中の AND-分離 $s = (s_1, s_2, \cdots, s_k)$ に対して、$s_i = (I_i, O_i, RS_i, IC_i, OC_i)$ で表される部分表現式が競合しない。すなわち、$O_i \cap (O_j \cup I_j \cup RS_j) = \phi (i \neq j, 1 \leq i, j \leq k)$ である。

(3) タスクの O 中のデータベースへ出力するデータが関わる述語を、タスクに関連する述語とする。各述語に関連するタスクは、順次演算で惟一に終端タスクが決められ、終端タスクでその述語の真偽を判断できる。

条件 (1) は、一貫したデータベースにおけるトランザクションの実行は、一貫性を保持することを要求している。条件 (2) と (3) は、WF で並列実行が可能であるためのものである。例えば、$WF_2 = (\{t_4, t_5\}, (t_4, t_5), x + y \leq 10)$ で、

$I_4 = \{x\}, IC_4 : \{x \leq 3\}, O_4 = \{y\}, OC_4 : \{y \leq 6\}$

$I_5 = \{y\}, IC_5 : \{y \leq 3\}, O_5 = \{x\}, OC_5 : \{x \leq 6\}$

は条件 (2) を満たさない。$x = y = 3$ という場合で実行されると、結果のデータベースは一貫性を満足しない。

妥当でないワークフロー $WF = (\Sigma, r, C)$ は、データベースの一貫性を保持するトランザクションに対して入出力データが等しいトランザクションが存在するような妥当なワークフロー WF に変換できる。

実行可能なワークフローに対して、それぞれの場合についての 1 つの交換方法を次に示す。

- 条件 (1) に対しは、$r{\cdot}a$ という形で r の出力データ O 中の出力条件 OC を満たさない部分 $O - O'$ を最終的に出力しない事後処理 $a = (O, O', \phi, OC, C)$ を設ける。
- 条件 (2) に対しは、AND-分離中の変更同士による競合タスクを 1 つにまとめ、検索と変更による競合タスクは、検索するタスクを抽出し、AND-分離と順次実行として定義し直す。
- 同じ述語に関わるデータを出力しているタスクの順次関連上での終端を、その述語の対応する終端タスクという。終端タスクで述語の真偽を判断できるために必要なデータ項目を入力データ項目に含む。条件 (2) より、それらのタスクは競合することになる。したがって、終端タスクは惟一に存在しかつ述語の真偽を判断できる。

このため、以降、妥当なワークフロー WF に限って議論する。

4.3 トランザクションの性質

本節では、一貫したデータベースで単独実行しているトランザクション中の各タスクの入出力データが満たす性質について考察する。

まず、タスクの入力データ $I \cup RS$ について分析する。$t_1 \cdot t_2$ について、t_1 を t_2 の親 (t_2 を t_1 の子) とする。$r = (t_1, t_2, \cdots, t_k)$ について、r の親を $t_i (i = 1, 2, \cdots, k)$ の親 (t_i を r の親の子 i) とする。さらに、祖先の入力条件の中、終了するときにも満たすものを、t の推移的な入力条件と呼ぶ。

各タスクの入力データ $I \cup RS$ には、祖先の出力データ、祖先の出力ではないが祖先の入力データであるもの、外部からのデータという 3 種類がある。トランザクションが単独実行されるときには、前者 2 種類のデータは他のトランザクションに変更されることなく、そのまま子表現に渡される (図 5)。このため、次のような性質を満たす。

補題 1 一貫したデータベースに単独実行しているトランザクション中の各タスクのインスタンス α の入力データ $I \cup RS$ は、推移的な入力条件を満たす。

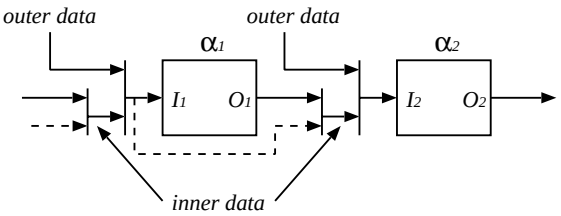

図 5 入力データの種類

証明： 帰納法で証明する。まず、実行可能なワークフローにおいて入力データが入力条件 IC を満たす。

WT の表現式には、順次と AND-分離という 2 種類の演算がある。演算個数が i 以上の場合に成り立つとする。

演算個数が i の場合は、次のように分けられる。

- 順次 $\alpha_1 \cdot \alpha_2$

  α_2 中のの内部入力データは、1) 親の出力データ $(I_2 \cup RS_2) \cap O_1$、2) 親の入力データ $((I_2 \cup RS_2) - O_1) \cap (I_1 \cup RS_1)$、3) 祖先の入力データ $((I_2 \cup RS_2) - O_1) - (I_1 \cup RS_1)$ という 3 種類に分けられる。3) の祖先の入力データは、祖先入力条件を満たす。2) の親の入力データは、親入力条件 IC_1 を満たす。したがって、内部からの入力データは、推移的な入力条件を満たす。
- AND-分離 $(\alpha_1, \alpha_2, \cdots, \alpha_k)$

  $(\alpha_1, \alpha_2, \cdots, \alpha_k)$ の入力データ $I \cup RS$ 中の内部からの入力データが、推移的入力条件を満たすため、$I_i \cup RS_i \subseteq I \cup RS (i = 1, 2, \cdots, k)$ 中の内部からの入力データも推移的入力条件を満たす。

このため、補題が成り立つ。 □

例 6 例 5において、α_2 の推移的な入力条件は、α_1 の入力条件「IC_1:$x_4 \leq 0$」である。

節内の述語に関連する終端タスクの順次演算上の終端を、節に関連する終端タスクという。

定理 1 一貫したデータベースに単独実行しているトランザクションは、節に関連する終端タスクのインスタンスがその節を満たす。

証明： 終端タスクのインスタンス α_i で節 c_j 満たさないとする。すなわち、$O_j \cup (I_i \cup RS_i - O_i)$ と入力条件 IC_i 推移的な入力条件で c_j を満たすと判定できない。定義 4、定義 7と終端タスクの意味より、それらのデータ項目の値はトランザクションの c_j の判定にも用いられている。このため、トランザクションの実行結果が c_j を満たすとは判定できないことになる。

妥当なワークフローの定義と矛盾する。 □

5. 並行実行の正当性

本章では、単独実行時のトランザクションと同様な性質を満たすという方針の下で並行実行時の正当性を導入する。

定義 9 Σ 上のトランザクション $WT_i(NT_i,ET_i)$ 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。

- $HN=\bigcup_i NT_i$,
- $HE \supseteq \bigcup_i ET_i$
- 異なる処理単位のタスクが競合するなら、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。

例 7 例 1において、同一クライアントが同時に 2 枚のクレジットを申請するスケジュールは、図 7で示されるようなグラフになる。

$$\begin{array}{ccccc} \alpha_0^1 & \rightarrow & \alpha_1^1 & \rightarrow & \alpha_2^1 \\ \downarrow & & \downarrow & & \downarrow \\ \alpha_0^2 & \rightarrow & \alpha_1^2 & \rightarrow & \alpha_2^2 \end{array}$$

図 6 スケジュール WH_1

α_2^2 が実行されるとき、推移的な入力条件 $x_4 \leq 0$ を満たさない。

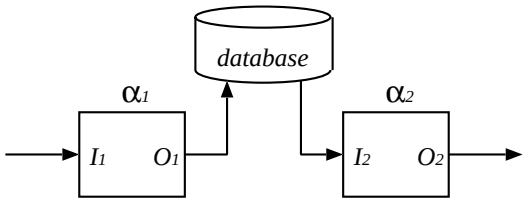

図 7 並行実行時のデータの流れ

並行実行時に、祖先の出力データまたは祖先の入力データである内部データは、他のトランザクションに変更されることがある。このため、図 7で示される並行実行時のデータの流れの下で、正当性基準を検討する。$t_0(\phi,DB,\phi,C)/t_f(DB,\phi,C,\phi)$ というすべての項目を出力/入力する特別なタスクがあるとする。単独実行と同じ性質を満たす方針で並行実行の正当性を定義する。

定義 10 スケジュール $t_0\ WH\ t_f$ が、次の条件を満たすとき**正当**であるという。各タスクのインスタンス $\alpha \in WT \in WH \cup \{t_0\}$ が開始するときに、α の入力データが入力条件 IC を満たし、かつ推移的な入力条件も満たす。

まず、これは、従来の直列可能性というような競合操作間の実行順序による基準ではなく、制約に基づく基準であることが分かる。例えば、各タスクのインスタンス α の入力データ中の内部データは、入力条件と推移的な入力条件を満たす範囲内での他の処理単位による変更は許可できる。ただし、H_1 における α_2^2 が推移的な入力条件を満たさないため、H_1 は正当なスケジュールでない。これで推移的な入力条件の必要性が示される。

定理 2 正当なスケジュール $t_0\ WH\ t_f$ において、t_f は入力条件を満たす。すなわち、WH が終了する時のデータベースは一貫性制約を満たす。

証明： 一貫性制約 C 中の各節 c に対して、c が 1 つの WT のタスクに関連する場合には、妥当なワークフローの条件より満たす。

c が複数個の WT_i に関連する場合に、定理 1より、それぞれの WT_i において節 c と関連する終端タスクで c を満たす。すなわち、終端タスクの推移的な入力条件、入出力条件で c を満たすと保証できる。正当なスケジュールの定義より、$t_0\ WH$ においても各タスクが同じような推移的な入力条件、入力条件を満たす。出力条件は入力条件によって保証されるので、正当なスケジュール $t_0\ WH$ においても、それぞれの WT_i において節 c と関連する終端タスクで c を満たす。このため、定理が成り立つ。 □

ここで、正当なクラスの性質について簡単に考えてみよう。データ項目の値を考慮した上で並行実行の制御を行う方法は効率的である [9]。ただし、それには、トランザクションをどのように細分して考慮するかという問題がある。個別のタスクの実行結果が要求されている制約を満たさなくても、トランザクション全体の実行結果がそれを満たすことがある。現時点までのタスクの実行結果が要求されている制約を満たしていても、トランザクション全体の実行結果がそれを満たさないこともある。したがって、全般的に考慮した方が効果的である。本論文が導入したトランザクションの再帰的な記述方法は、そのための基礎を与えている。

6. 関連研究との比較

並行処理制御基準は、大きく次の 2 種類に分けられる。

(1) どのデータ項目にどのような操作を行うかによる制御。操作結果は考慮しない。

(2) 操作結果がデータ項目に課している条件 (制約) を満たすかどうかによる制御。

従来の直列可能性基準に基づく方式[4]は前者の例である。論文[9]は、並行処理制御には、従来ののデータベースのあらゆるデータ項目の値に適用できる方式を用いるのではなく、制約に基づく制御方式の必要性と有効性を訴えている。

本論文が提案している基準はは後者に属するので、従来の直列可能性を拡張する方式と異なる。また、単独実行の結果と等価する方法で正当性を提案しているので、直列実行と等価となるスケジュールの範囲を拡大する方式[2]などとも発想が異なる。

ワークフローの並行実行時の制御について、データベースの研究者は、何種類もの拡張されたトランザクションモデルを提案してた[5]。しかし、それだけでは、ワークフローの応用分野の要求に答えられなかった[7]。そのため、データベースとワークフローの両側が互いに智慧を借り、トランザクショナルワークフローという概念が生まれた。

広く受けいられているワークフローの並行処理の正当性基準が存在しないという問題点が、論文[3]より指摘されている。そのため、論文[3]では、ワークフローを形式的に定義し、制約に対する施錠を行う並行処理制御方式を提案している。ただし、ハイパーグラフを用いて制御フローを表しているため、記述が複雑である上、反復処理も表せない。並行実行における正当性基準は、制約を満たさない間施錠を行うという方針の下で与えているので、本論文とは異なる基準となっている。また、制約を満たすかどうかは各タスク単位で考慮しており、推移的な入力条件を考慮していない。

7. むすび

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、ワークフローのモデル化を行った。制御フローに対応する表現式を用いることでトランザクションの定義を簡潔にすることができた。また、ワークフローの実行可能性や出力データの性質についても検討した。トランザクション内の各タスクが単独実行時と同じ条件を満たす方針の下で並行実行の正当性基準を導入した。それに際して、各タスクの入出力条件という明示的な条件のみでは不充分である問題点を指摘し、推移的な入力条件も考慮しなければならないことを示した。

参考文献

1) Alonso, G., Agrawal, D., Abbadi, A. E., 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., 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) Elmagarmid, A. K. (ed.): *Database Transaction Models for Advanced Applications*, Morgan Kaufmann, 1992.

6) Leymann. F., and Roller. D. Business Process Management with FlowMark. Proceedings of 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) 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).

9) 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, 18, pp.4-10 (1995).

10) Puustjarvi, J. Tirri, H. Veijalainen, J.: Concurrency control for overlapping and cooperative workflows. IEEE Transactions on Computer Systems Bulletin pp. 24-30 (1996).