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

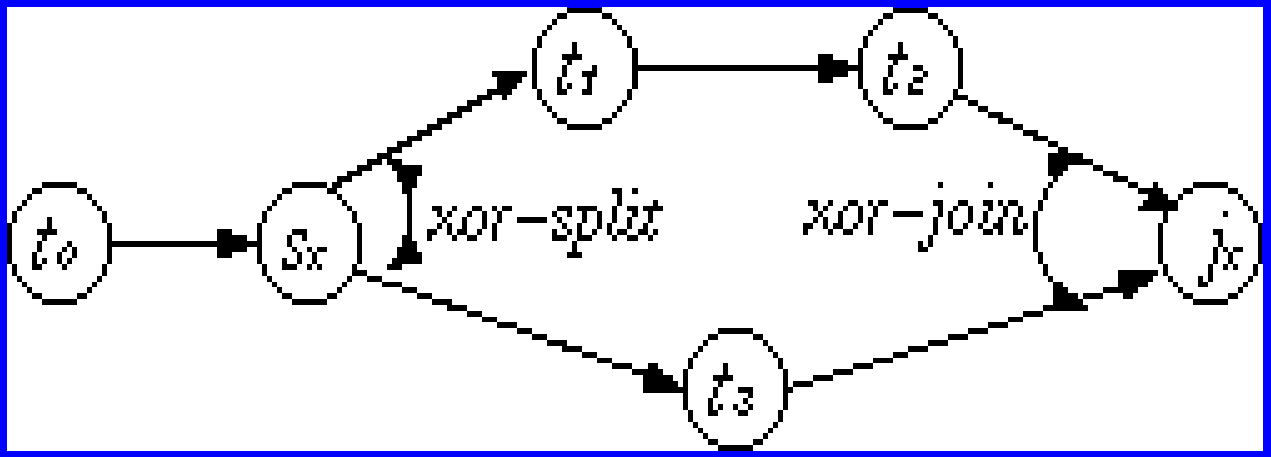
徐　海燕　　古川哲也　　史　一華
福岡工業大学　九州大学　西南学院大学

ワークフローにおける並行処理制御の必要性

クレジットカードの申請業務

- タスク
- 制御フロー

新規申請　　審査



受け付け

上限引き上げ

2枚同時に申請、使用できる額が倍に！

ワークフローにおける並行処理制御の課題

- 並行処理制御
  - ✧ 競合操作の順序
  - ✧ 意味論上単独実行と等価 ←有効

e.g.旅行会社の切符購入
便・日期の変更
キャンセル

- ワークフローのモデル化
  - ➢ 制御フロー（反復業務）？
  - ➢ トランザクション？
  - ➢ DBの一貫性制約とタスクの入出力条件の関係

並行実行の正当性基準の分類

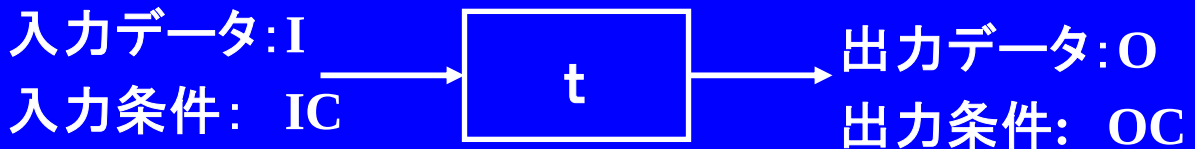
◆操作された値を考慮せず、競合操作の順序

- 直列可能性
- トランザクション階層
- ……

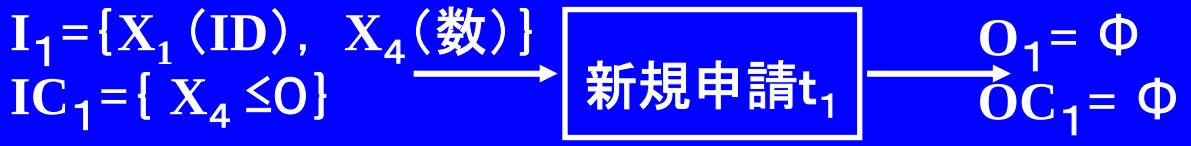
◆操作された値を考慮し、意味論 上単独実行と等価

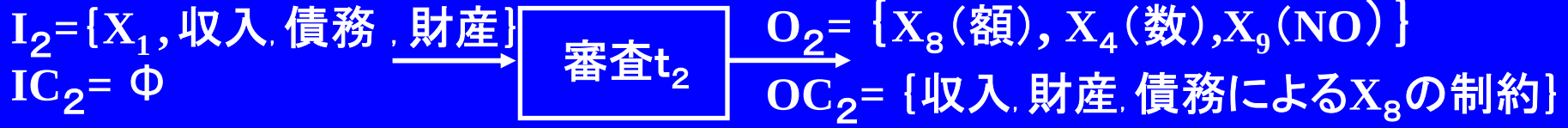
ワークフローのモデル化

- タスクt＝(I, O, IC, OC)　並行実行時に、原子性の単位

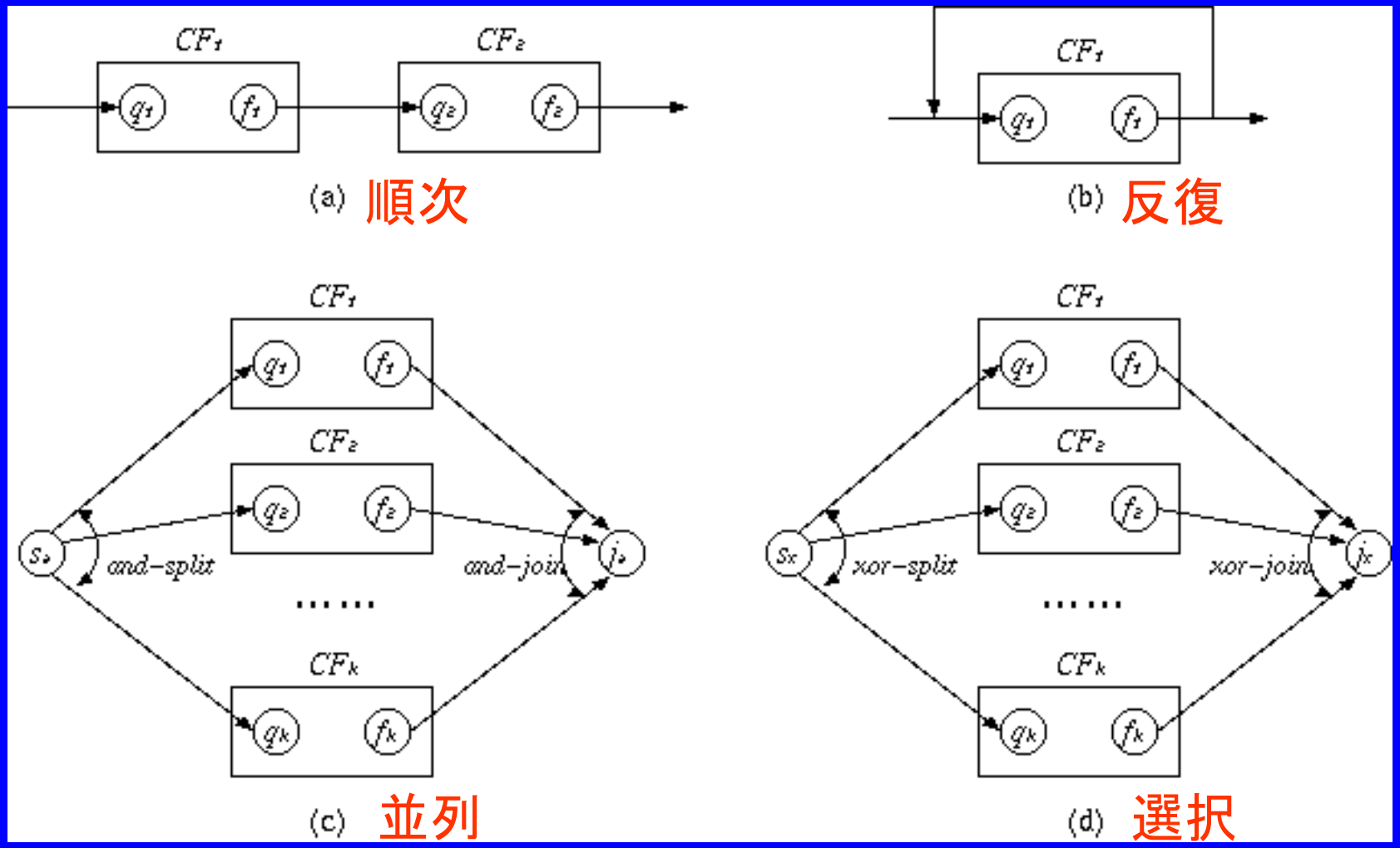


例:





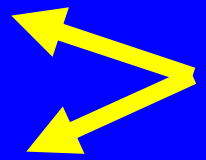
制御フローCF


CF1
CF2
q1
f1
q2
f2
(a) 順次
CF1
q1
f1
(b) 反復
CF1
CF2
CFk
sa
and-split
and-join
ja
(c) 並列
CF1
CF2
CFk
sx
xor-split
xor-join
jx
(d) 選択

ワークフローの妥当性

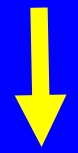
ワークフロー

トランザクション：インスタンス



ワークフロー内のすべてのトランザクション

- 実行可能性 （入力条件を満たす）
- 終了可能性 （結果DBの一貫性）
- 並列実行の正しさ （AND-分離内の実行が競合しない）



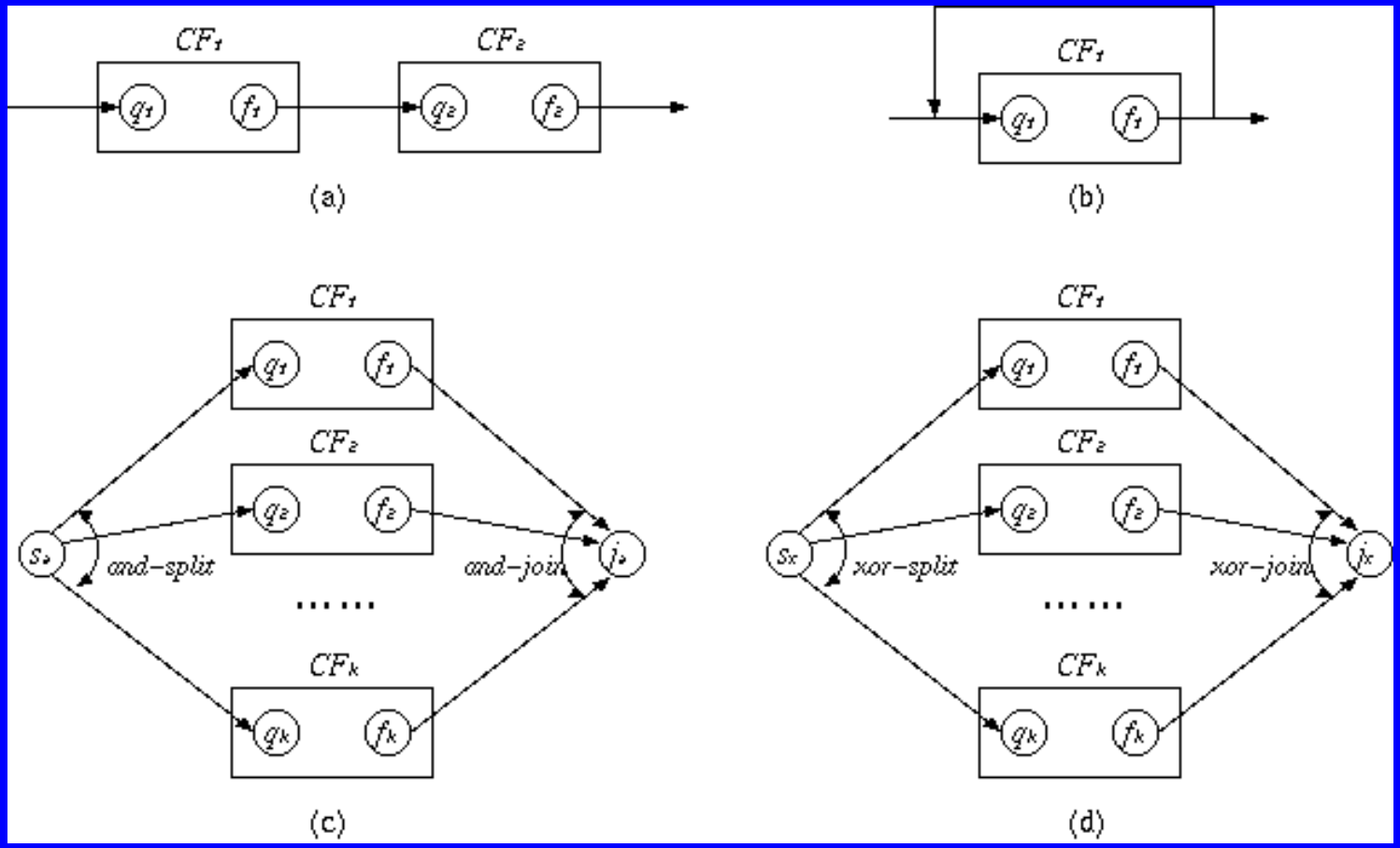
インスタンスレベルではなく

表現式レベルでの入出力データ、入出力条件で検討

制御フローCFと表現式r

$r_1 \cdot r_2$

r_1^+

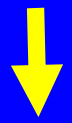


$(r_1, r_2, \cdots, r_k)$

$r_1 \oplus r_2 \oplus \cdots \oplus r_k$

表現式rの表す集合

表現式	集合
タスク t:	{t}
順次 $r \cdot s$:	{RS}
反復 r^+:	R^+
選択 $r_1 \oplus r_2 \oplus \cdots \oplus r_k$:	$\cup R_i$
並列 $(r_1, r_2, \cdots, r_k)$:	$R_1 \times R_2 \times \cdots \times R_k$



ワークフロー WF=(Σ, CF, C)

WF=(Σ, r, C)

DBの一貫性制約

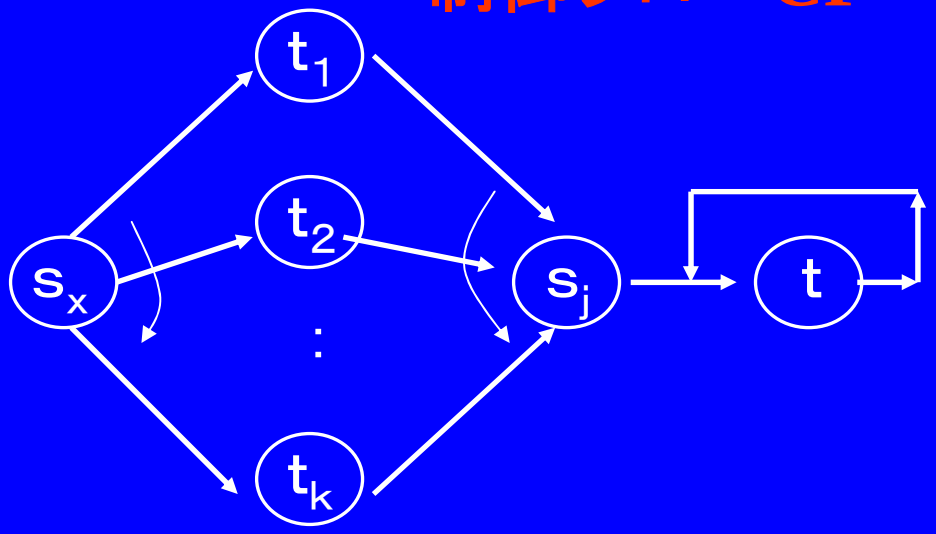
トランザクション

表現式rのRの要素のインスタンス

表現式r

$(t_1 \oplus t_2 \oplus \cdots \oplus t_k) \cdot t^+$

制御フローCF



集合$R=\{t_1 \cdot t,\ t_1 \cdot t \cdot t,\ t_2 \cdot t \cdot t \cdot t,\ \cdots\}$

トランザクションの表現式

表現式rのRの要素

- 選択と反復演算を含まない表現式
- 順次と並列演算の表現式

トランザクションの表現式の

- 入力データ、入力条件
- 出力データ、出力条件

を再帰的に計算できる

⇩

ワークフローの妥当性

入出力条件･一貫性制約の記述方法

- 節

$$A_1,A_2,\cdots,A_k \rightarrow B_1,B_2,\cdots,B_n$$

- 述語

A_i, B_j　例えば　$x+y>10$

トランザクションWTの表現式rの

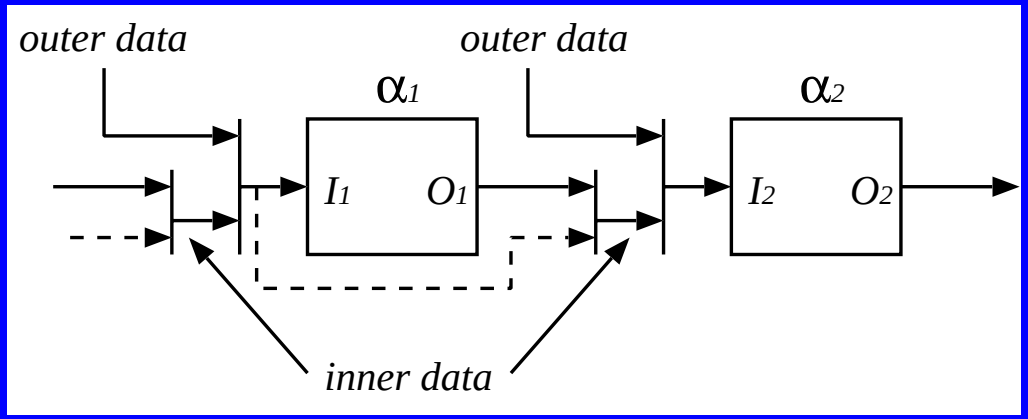
入力データI，入力条件IC，出力データO，出力条件OC

順次r・s：　$I=\mathbf{I_1}\cup(I_2-O_1)$

$IC=\mathbf{IC_1}\wedge IC_2$中の$(I_2-O_1)$のみに関わる節

$O=\mathbf{O_2}\cup(O_1-O_2)$

$OC=\mathbf{OC_2}\wedge OC_1$中$(O_1-O_2)$のみに関わる節



選択$(r_1,r_2,\cdots,r_k)$：　$I=\cup I_i$　　$O=\cup O_i$

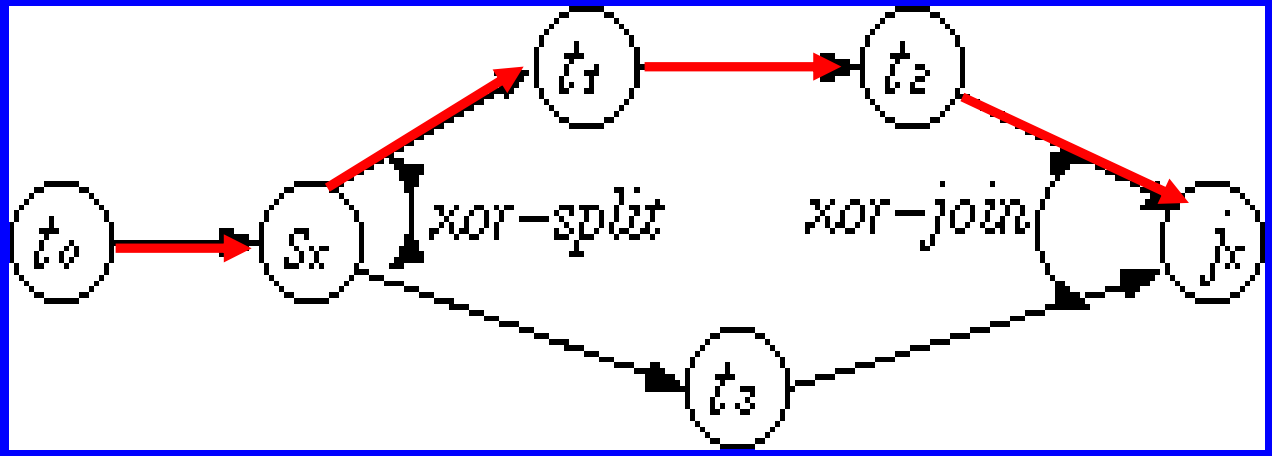
$IC=\wedge IC_i$　　$OC=\wedge OC_i$

例

WT_1:t_0 t_1 t_2のインスタンス：
入力条件IC_1、出力条件OC_2

新規申請　　審査

受け
付け



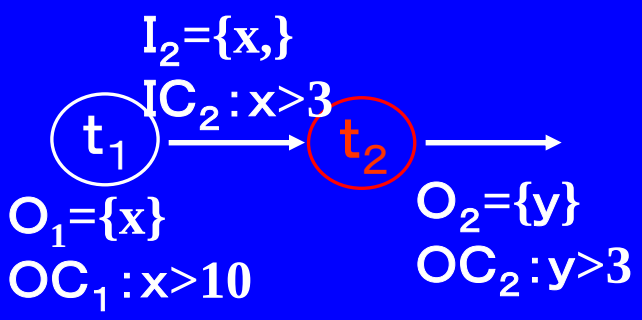
上限引き上げ

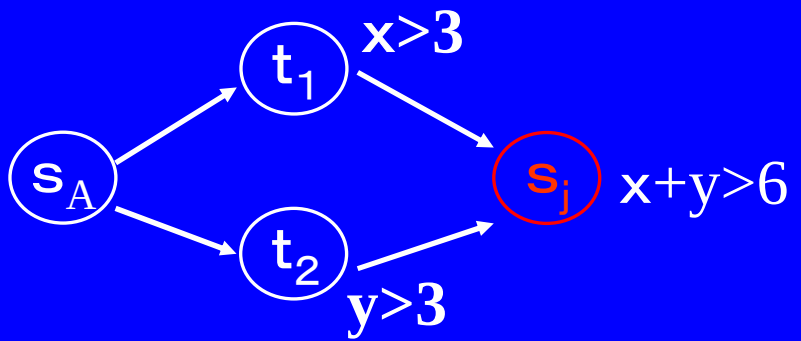
WT_2:t_0 t_3 のインスタンス：入力条件$\neg IC_1$、出力条件OC_3

結果DBの一貫性

WTの入力条件と出力条件で一貫性制約を保証

述語の関わるタスク：タスクの出力データの関わる述語





各述語・節の関わるタスクは、唯一の終端タスクが存在

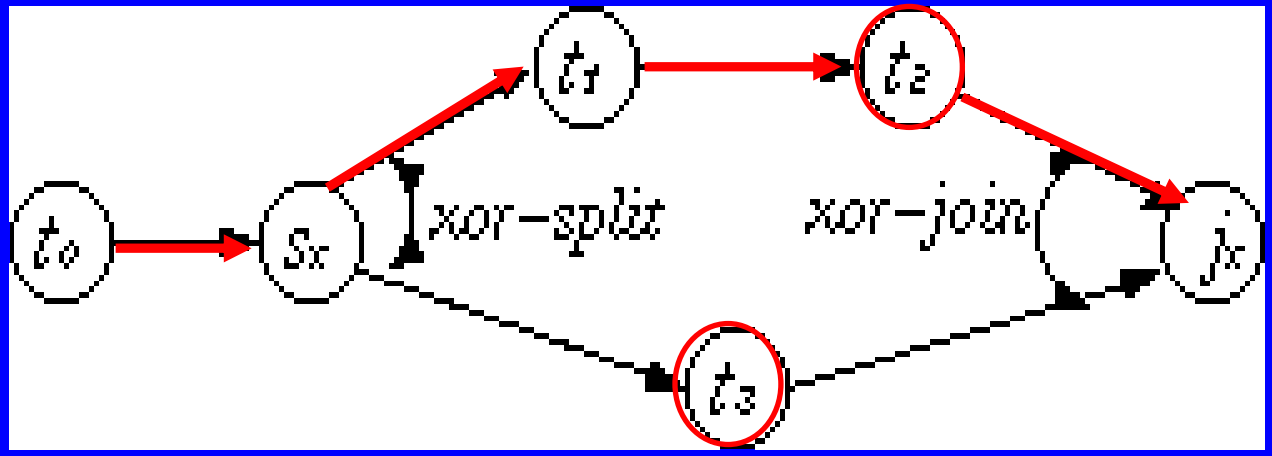
入力条件、推移的な入力条件、出力条件で節を保証

例

t_2の推移的な入力条件IC_1、出力条件OC_2 ⇒ 一貫性制約C

新規申請　　審査

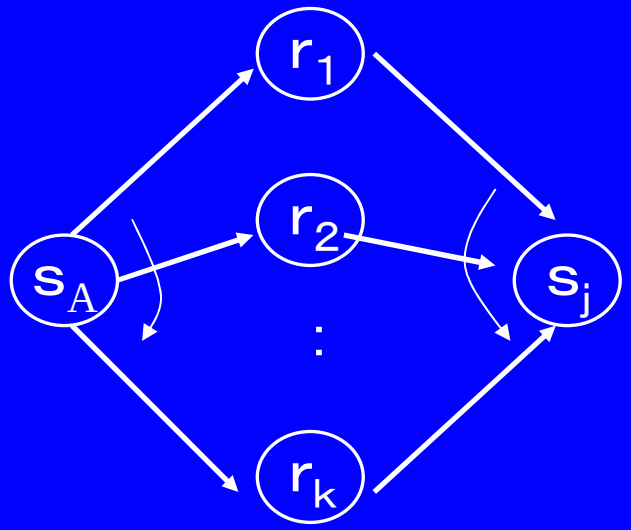
受け付け



上限引き上げ

¬IC_1、OC_3 ⇒ 一貫性制約C

並列実行：AND-分離



r_iの出力データは他のr_jの入出力データと競合しない

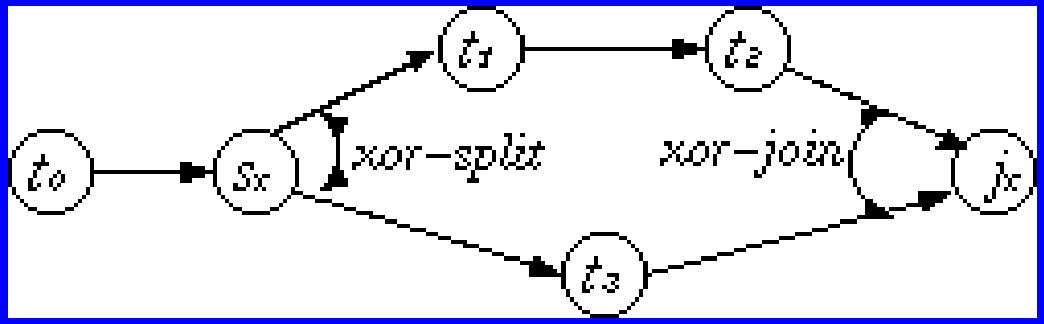
実行可能性

DBの一貫性制約Cを満たす

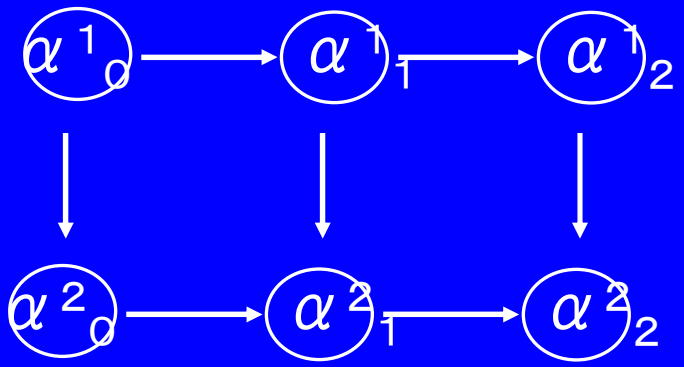
トランザクションWTの表現式rの入力条件を満たす
（r⊕r‘）

各タスクの入力条件を満たす

スケジュール



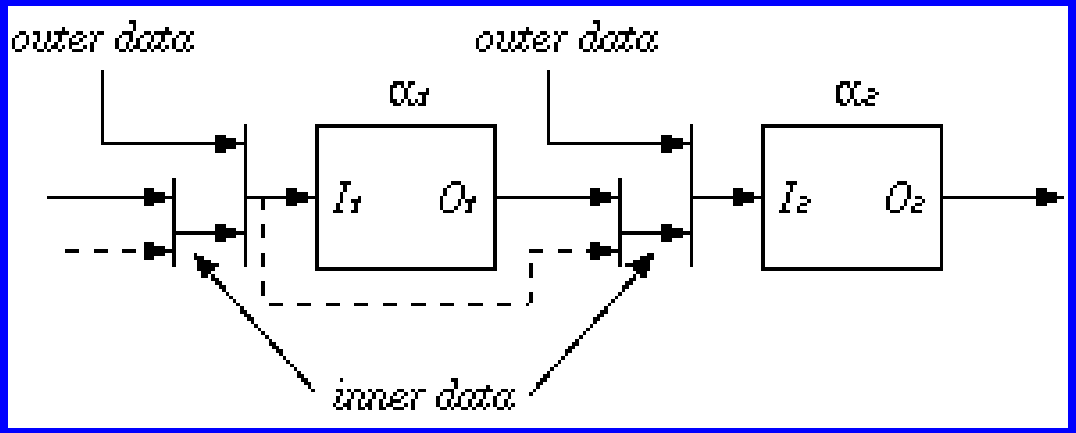
ワークフロー



トランザクション

スケジュール：
競合操作間の順序が定めた非巡回グラフ

単独実行時各タスクの入力データが満たす条件



祖先の入力データ → 推移的入力条件

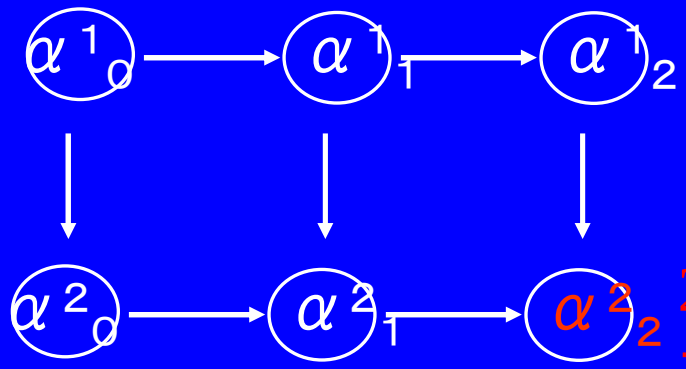
祖先の出力データ
データベースからのデータ
} 入力条件

並行実行の正当性基準

各タスクの入力データが

- 入力条件を満たす → 出力条件
- 推移的入力条件を満たす

DBの一貫性



2枚同時に申請すると、
推移的な入力条件を満たさない

まとめ

- ワークフローモデルの提案

  表現式による記述を導入

  - 反復作業を表せるワークフローのトランザクション
  - トランザクションの性質

- 並行実行における正当性基準の提案

  推移的な入力条件の必要性を指摘

今後の課題

- ワークフローの設計
- 並行処理制御方式の検討
- シミュレーション