ワークフローペトリネットによる
キメ細かな e-Learning コースウェア

大矢野 潤

1 はじめに

近年、大学教育におけるリメディアル教育の重要性が増している。リメディアル教育の目的とは、大学の授業を円滑に進めるために (1) 高等学校までの学習を補うもの (2) 高等学校で重点的に教わらなかったものの、特定の専門ではその理解が必須とされるもの (3) 就職支援などがあげられる。具体的には、(1) の読み書きや数学の補習に始まり (2) の、例えば情報分野でもとめられる N 進数変換、集合論の基礎やブール演算を暗黙裡に使用した推論が、 (3) においては、学力自体というよりもいわゆる「受験慣れ」をしていない学生のための就職試験対策などである。

リメディアル教育の背景としては、いわゆる「ゆとり教育」での履修科目数、授業時間数の減少や少子化による 18 歳人口の減少などにともなう大学生の学力低下があげられるが、メディアル教育実施時には、全体としての学力低下に加え個々の学生の理解度のばらつきも大きな障壁となる。理解度が十分でない学生には、問題の意図、問題文中で注目すべき箇所や使う公式など事細かに指導する必要がある一方、ある程度理解している学生には親切な指導よりもたくさんの問題を与えることが有効であることが多い。また、書き込みの多い「親切な教科書」の使用は、書き込みが学習者の理解の傾向に沿っている場合には効果的であるが、そうでない場合には、単なる情報過多を引き起こす。筆者は、このような経験から、学生の理解度に応じて指導の「キメの細かさ」を変えることのできる e-Learning システムが必要であると感じてきた。

e-Learning 学習教材の開発にあたっては、「問題作成」、「解説の仕方の工夫」、「実装」を同時にこなす必要があり、学習教材開発のさまたげとなっている。さらに、それぞれの学生に合わせた「キメ細かな」指導や解説を直接プログラミングにより実装した場合には、膨大な数の条件分岐の制御が必要となり、現実的な作業量を超えてしまう。このことに対処するため、e-Learning 教材のコースをペトリネットの一種であるワークフローネットとして実装することを提案する。

ペトリネットはその表現力、記述力から並行プロセスや工業プラントの制御システムなどに利用されている数学モデルであり、コース記述目的としては十分すぎるほどの能力を持つ。本システムでは、コース設計者は直接ペトリネットを定義するのではなく、XML 形式の簡易ワークフローネット言語を用いてコースを記述する。この記述は、変換ツールにより JavaScript 形式のペトリネットへと変換される。このペトリネットとペトリネット解釈ライブラリ、HTML5 による画面定義を組み合わせたものが最終的な e-Learning 教材となる。コース設計者は HTML5 による「教材の見た目」の定義と XML による「コースの記述」に専念することができ、煩雑なプログラミング作業をスキップできる。

本研究で行ったことには次のものがある。

- e-Learning コースウェアをワークフローネットとして記述することを提案した

- ワークフローネットの構築子を定義し、形式的意味を与えた
- ワークフローネットの XML 文法を定義した
- ワークフローネットを JavaScript 形式のペトリネットに変換するプログラムを開発した
- ペトリネットの解釈、実行エンジンを開発した
- 具体例として「N 進数変換コース」を作成した

なお、ここでは単独の学習者を想定しており、複数の学習者や学習を管理する仕組み (LMS: Learning Management System) は本論文の範囲外である。これらの課題に対応するためのペトリネットの高レベル化 (時間ペトリネット、カラーペトリネット) や階層化の開発は今後の課題としている。

本論文の構成は次のようになっている。第 2 章で e-Learning システムを簡単に紹介する。つづいて、第 3 章でペトリネットとそのサブクラスであるワークフローネットとその構築子の図的、形式的な意味づけを行う。第 4 章では、ワークフローネットに基づいたコース定義言語、XML 文法での記述方法やユーティリティーを定義する。第 5 章で N 進法に関するコースの実際の記述と実行例を紹介し、最後に考察と今後の課題について述べる。

2 e-Learning システム

e-Learning システムとは電子的なデバイスなどを利用した学習支援システムのことをいい、「教材・学習材」と「学習管理システム (LMS)」から構成される。近年、情報通信技術 (ICT: Information and Communication Technologies) の発達と普及にともない、電子メールや掲示板、SNS による学習者支援、マルチメディアを駆使した教材の開発が盛んである。利用されるデバイスの代表的なものには、パーソナルコンピュータ、携帯端末、携帯電話、スマートフォンそして CD/DVD などの AV 機器などがある。e-Learning システムは必然的にデバイス、メディア、情報通信技術の発達と深いかかわりをもつ。

以下、e-Learning システムの発達過程を簡単に振り返ってみよう。なお、e-Learning システムの歴史、理論については [4] に詳しい解説がある。

2.1 e-Learning システムの歴史

2.1.1 Computer Assisted Learning/Computer Aided Instruction

「e-Learning」という言葉はインターネットの普及とともに発明された言葉であるが、いわゆる IT 革命以前から CAL (Computer Assisted Learning) や CAI (Computer Aided Instruction)、すなわちコンピュータによる学習補助という試みが盛んにおこなわれてきた。学習教材メディアとしてはフロッピーディスクや磁気テープ、MO や CD などの光学メディアか使用され、物理的に配布されてきた。フロッピーディスクでは汚れや経年変化によるデータの劣化や損失が、CD/DVD ではメディア配送にかかる金銭的、時間的コストが問題となってきた。加えて、これらの教材の配布は一方向であり、学習者の質問に実時間で応えることなどは想定されていない。

2.1.2 Computer Mediated Communication

一方向コミュニケーションの問題を解決する手段として、1990 年前後から急速に復旧したインターネットを利用した CMC (Computer Mediated Communication) が提案されてきた。これは、メールや電子掲示板などの他の学習者とのコミュニケーションを図る仕組みを積極的に利用し、教室での対面授業に近い環境を作り出そうとする試みである。CMC は「同期型 CMC」と「非同期型 CMC」の二つに分類され、前者では、チャットやビデオ会議、後者では電子メールや掲示板、ブログなどが利用されている。

2.1.3 Web Based Learning

インターネットの普及に寄与した初期のキラーアプリケーションはファイル転送プログラム FTP と電子メールである。これらはあくまでインターネットを利用した個別のアプリケーションであり、つづいて、これらのインターネットアプリケーションを統一的に利用する機能をもつ WWW (World Wide Web) が登場する。日本では最初のホームページが 1992 年に作成され、以来、ユーザフレンドリーなインターフェイスとその使い勝手の良さから、爆発的に世界中に普及した。WWW の e-Learning システムへ活用は、CAL、CAI、CBT (Computer Based Training) などの自主学習教材を WBL (Web Based Learning)、WBT (Web Based Training) へと進化させた。教材や学習者の成績が Web サーバ上で管理されることにより、実時間的、総合的な学習状況の把握が可能となる。さらに同じく WWW 上で展開されている SNS (Social Networking Services) などと連携することで新たな学習環境の構築が期待されている。

2.1.4 e-Learning システムのまとめ

e-Learning システムは、CAL/CAI/CBT など自主学習を主とするものと Web サービス、特に SNS などと融合した CMC 機能を持つ WBL/WBT が開発されている。インターネット上でこれらのサービスを展開する利点としては、学習教材の配布、アップデートがスムーズに行われるということ、孤立しがちな学習者が他者とコミュニケーションをとることにより相互にエンカレッジすることが期待できるということがある。

加えて、学習管理システム LMS は e-Learning システム導入における強力な動機の一つである。学習教材管理・提供と学習者の学習の学習状況の把握は、学習者へのより効果的な指導に欠かすことができない。

2.2 e-Learning システムの利点と欠点

e-Learning システムは決して万能ではなく、利用すればだれでも効果的に学習ができるというものではない。ここでは、現在指摘されている利点と欠点について、簡単に箇条書きにする。

利点:

- 同時間、同一場所に集まる必要がなく、自由な時間と場所で学習できる
- 個人の習熟度に応じて学習を進めることができる
- 目的に応じて標準化された授業を受けることができる
- 印刷教材の量を減らすことができる

欠点:

- 学習意欲の持続が難しい
- 質疑などによる実時間での問題解決がしにくい
- 教師やほかの学習者とのコミュニケーションを図ることが難しい
- 学習者のドロップアウト率が高い

これらは、個人利用の観点からの見解であり、大学授業の「補助教材」として用いるケースとは背景が異なることに注目したい。大学の教室では、教員 – 学生、学生 – 学生間のコミュニケーションは「対面」により確保されている。さらに、同じ時間に同じ教室内で受講している学生間のばらつきがあるケースで単一教材を使用する時には、解き方の解説や問題数などの調整を行う必要がある。ある学生には退屈極まりなく、別の学生には速すぎて追いつけない教材になってしまうからである。

大学におけるリメディアル教育の補助教材としての e-Learning システムには、同一内容であっても、学習者の理解度に対応したバリエーションをつけることができるコースづくりを支援する環境が必要なのである。

3 ペトリネット

ペトリネット (Petri Net) は、1962 年に Carl Adam Petri によって発表された図的で数学的なモデリングツールである。その応用範囲は広く、並行 (concurrent) 、並列 (parallel)、非同期 (asynchronous)、分散 (distributed)、非決定的 (non-deterministic)、そして確率的 (stochastic) な性質をもつシステムを形式的に記述できる。加えて、生来の図的な表現とともに、対象システムを直観的かつ数学的に解析する手段を提供する。ペトリネットは工業プラント制御システムや通信プロトコル、マルチプロセッサの解析などの分野で成功しており、近年では WWW の発達に伴い、ビジネスプロセス解析などへの応用が始まっている [2, 3]。

ペトリネットの研究は依然活発に行われており、さまざまな解析アルゴリズムやツールが開発されている。研究成果はよく整理されたサーベイ論文 [1, 5] や教科書 [6] の形で利用することができる。本研究において使用している定義やさまざまな結果は [1, 2, 5] から引用した。

3.1 ペトリネットの定義

定義 3.1 (ペトリネット) ペトリネット *(PN)* とは

$$PN = (N, M_0) = (P, T, F, W, M_0)$$

であらわされる *5* 項組である。ここで、P はプレースの有限集合、T はトランジションの有限集合であり、$P \cap T = \emptyset$ とする。また、$F \subseteq (P \times T) \cup (T \times P)$ はアーク *(*フロー*)* の集合、$W : F \to \{0, 1, 2, \ldots\}$ はアークから非負整数への写像であり、アークの「重み」を意味している。さらに、マーキング $M : P \to \{0, 1, 2, \ldots\}$ はプレースから非負整数への写像であり、「マーキング M でプレース p は $M(p)$ 個のトークンをもつ」ことを表す。特に、M_0 は初期マーキングを意味する。

なお、複数のペトリネットについて議論する場合に、プレースやトランジションの集合がどのペトリネットのものなのかを明確に示す必要がある場合には、「ペトリネット名. メンバ」のように記述することにする。例えば、ペトリネット PN のプレースの集合が PN_1 、PN_2 それぞれのプレースの集合の和に等しい場合には、$PN.P = PN_1.P \cup PN_2.P$ のように記述する。

プレースの数が少ない時には、マーキング M をプレースに対応するトークンの数をならべることで表すことがある。例えば、$M(p_1) = 2, M(p_2) = 1, M(p_3) = 0$ のとき、単に $M = (2, 1, 0)$ もしくは、$M = 2p_1 + 1p_2 + 0p_3$ と記述する。

本論文では、W の値域を $\{0, 1\}$ に制限して使用する。すなわち、$W(x, y) = 0$ は単に $(x, y) \notin F$ を、$W(x, y) = 1$ は $(x, y) \in F$ を意味する。 本研究における W の定義は F の定義と完全に同一視できるため特に必要がない場合には W の記述を省略する。

定義 3.2 (ペトリネットの状態) ペトリネットのマーク M と、記述されるシステムの「状態」を同一視することにより、システムの状態の変化をマークの変化により表現する。

定義 3.3 (入出力集合) プレース p、トランジション t に対して $(p, t) \in F$ であるとき、 p は t の入力プレース、t は p の出力トランジションであるという。また、アーク (p, t) は t の入力アーク、p の出力アークであるという。$(t, p) \in F$ であるとき、 p は t の出力プレース、t は p の入力トランジションであるという。同様にアーク (t, p) は t の出力アーク、p の入力アークであるという。

$\bullet t$ はトランジション t の入力プレースの集合、$t\bullet$ は t の出力プレースの集合を、$\bullet p$ はプレース p の入力トランジションの集合、$p\bullet$ は p の出力トランジションの集合をそれぞれ表現する。

定義 3.4 (発火可能) $\forall p \in P; M(p) \geq W(p, t)$ のとき、マーキング M でトランジション t が発火可能 *(Enable)* であるという。マーキング M でトランジション t が発火可能であるということを $M[t\rangle$ であらわす。

定義 3.5 (発火) $M[t\rangle$ であるとする。この時、トランジション t が発火 *(Fire)* した後のマークが M' であるということを $M[t\rangle M'$ と記述する。マーク M と M' の間には次の関係が成り立つ。

$$\forall p \in P; M'(p) = M(p) - W(p, t) + W(t, p)$$

なお、発火可能な可能なトランジションは発火してもしなくてもよい。

定義 3.6 (到達可能性) 状態 $M(= M_0)$ から状態 M_n が到達可能 *(Reachable)* であるとは、発火可能なトランジションの系列と状態の系列が存在することであり、次のように記述する。

$$M_0[t_0\rangle M_1[t_1\rangle \ldots [t_{n-1}\rangle M_n$$

もしくは、単に $M[*\rangle M_n$ と記述する。

定義 3.7 (活性) ペトリネット PN が活性 *(Live)* であるとは、初期状態から到達可能なすべての状態 M' とすべてのトランジション t に対し、$M'[*\rangle M''$ となる状態 M'' が存在し、 $M''[t\rangle$ とできることをいう。

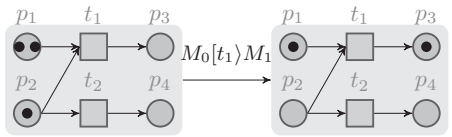
定義 3.8 (有界性、安全性) ペトリネット PN が有界 *(Bounded)* であるとは、すべてのプレース p に対し、ある自然数 n が存在し、すべての到達可能な状態 M について $M(p) < n$ となることである。ペトリネット PN が安全 *(Safe)* であるとは、すべてのプレースのトークンの最大数が高々*1* であることをいう。

定義 3.9 (パス) ペトリネット PN のパス *(Path)* C とは、$1 \leq i \leq k-1$ のすべての i について $(n_i, n_{i+1}) \in F$ となるノード $n_i \in P \cup T$ のシークエンス $(n_1, \ldots, n_k)$ のことである。

定義 3.10 (強連結) ペトリネット PN が強連結 *(Strongly Connected)* であるとは、すべてのノード $x, y \in P \cup T$ のペアについて x から y へのパスが存在することとする。

3.2 ペトリネットの例

例として、簡単なペトリネット PN を紹介する。ここで、PN は初期状態 (初期マーキング) M_0 でトランジション t_1、t_2 が発火可能であり ($M_0[t_1\rangle, M_0[t_2\rangle$)、それぞれの発火後は M_1、M_2 に状態遷移する ($M_0[t_1\rangle M_1, M_0[t_2\rangle M_2$) ことがわかる。図 1 では、$M_0[t_1\rangle M_1$ 遷移を選択している。なお、 M_1, M_2 それぞれの状態で発火可能なトランジションは存在しない ($\nexists t \in T; M_1[t\rangle$ or $M_2[t\rangle$)。



$$
\begin{aligned}
PN &= (P, T, F, M_0) \\
P &= \{p_1, p_2, p_3, p_4\} \\
T &= \{t_1, t_2\} \\
F &= \{(p_1, t_1), (p_2, t_1), (p_2, t_2), \\
&\quad (t_1, p_3), (t_2, p_4)\} \\
M_0 &= 2p_1 + 1p_2 + 0p_3 + 0p_4 \\
M_1 &= 1p_1 + 0p_2 + 1p_3 + 0p_4 \\
M_2 &= 2p_1 + 0p_2 + 0p_3 + 1p_4
\end{aligned}
$$

図 1: ペトリネット遷移の図的表現とその定義

3.3 ペトリネットのサブクラス

ペトリネットの有効性はその記述能力の高さにある。半面、本質的に解析不可能、もしくは極端に大きな計算量を必要とする問題も容易に記述できてしまうため、無制限に記述されたペトリネットに対して可能問題や活性問題を直接解析することは現実的でない。このため、いくつかの制限を加えたペトリネットのサブクラスが研究されており、その解析に必要な計算量も明らかになってきている。例えば、マークグラフ (MG)、状態機械 (SM)、無競合 (CF) ネット、自由選択 (FC) ネット、拡張自由選択 (EFC) ネットをはじめとし、本研究で用いるワークフローネット (WorkFlow Net) についても有効な研究がなされている [5]。これらの結果を踏まえ、利用者それぞれの使用目的に合わせて適切なペトリネットのサブクラスを選択するのが一般的である。

3.4 ワークフローネット

ここでは、ペトリネットのサブクラスのひとつであるワークフローネットを定義するとともに、ワークフローネットが満たすべき性質について簡単に紹介する。詳しくは [2] を参照されたい。

定義 3.11 (ワークフローネット) ペトリネット PN がワークフローネット WF であるとは次のことをいう。

1. $WF(PN)$ は 2 つの特別な状態 p_i、p_o をもつ。ここで p_i は入力トランジションを持たないプレース *($\bullet p_i = \emptyset$、ソースプレース)* であり、p_o は出力トランジションを持たないプレース *($p_o\bullet = \emptyset$、シンクプレース)* である。

2. $WF(PN)$ に p_o と p_i をつなげるトランジション $\bar{t}$ *(*すなわち $\bullet\bar{t} = \{p_o\}$、$\bar{t}\bullet = \{p_i\}$*)* をつけ加えたペトリネットは強連結になる。

特にペトリネットがワークフローネットであることを強調する場合には、WF と書く。

定義 3.12 (健全性) ワークフローネット WF が健全 *(Sound)* であるとは、次のことをいう。

1. $\forall M; (M_0[*\rangle M) \Rightarrow (M[*\rangle M_f$ *s.t.* $M_f(p_o) = 1)$

2. $\forall M; (M_0[*\rangle M \wedge M(p_o) \geq 1) \Rightarrow ((\forall p \in P; p \neq p_o \Rightarrow M(p) = 0) \wedge M(p_o) = 1)$

3. $\forall t \in T; \exists M, M'; M_0[*\rangle M[t\rangle M'$

ワークフローネット $\overline{WF}$ により、WF を強連結にするトランジション $\bar{t}$ を加えたもの、すなわち $\overline{WF} = (WF.P, WF.T \cup \{\bar{t}\}, WF.F \cup \{(p_o, \bar{t}), (\bar{t}, p_i)\})$ と表す時、次の定理が成り立つ。

定理 3.1 (健全性) ワークフローネット WF は $\overline{WF}$ が活性で有界であるとき、その時に限り健全である *[2]*。

定理 3.2 (健全性の検証) 自由選択ワークフローネットの健全性は決定性多項式時間で検証可能である *[5]*。

3.5 ワークフローネット構築子

本研究では、e-Learning 教材のコースと特別な形のワークフローネットを同一視する。ここでは、実用的なコースを構築するために、小規模のワークフローネットから大規模かつ複雑なワークフローを構築するためのワークフローネット構築子を定義する。

定義 3.13 (ワークフローネット構築子) ワークフローネット構築子 *(WorkFlow Net Constructor)* は次の *BNF (Backus-Naur Form)* によって定義される。

$$
\begin{aligned}
WF \quad ::= \quad & t \in T \\
| \quad & \textbf{reference of } WF \\
| \quad & WF_1 \ \rightarrow \ WF_2 \\
| \quad & WF_1 \ \otimes \ WF_2 \\
| \quad & WF_1 \ \oplus \ WF_2
\end{aligned}
$$

すなわち、ワークフローネットはそれよりも小規模な「トランジション $t \in T$」、「参照 ***(Reference of)***」「逐次結合 *(*$\rightarrow$*)*」「並列結合 *(*$\otimes$*)*」「選択結合 *(*$\oplus$*)*」によって与えられる。

ここで次のことが成り立つ。

定理 3.3 (ワークフローネット構築子) ワークフロー WF_1, WF_2 が $WF_1.P \cap WF_2.P = \emptyset$ かつ $WF_1.T \cap WF_2.T = \emptyset$ の条件を満たすとする。このとき、ワークフローネット構築子によって合成されたペトリネットは再びワークフローネットとなる。すなわち、$WF_1 \to WF_2$、$WF_1 \otimes WF_2$、$WF_1 \oplus WF_2$ は再びワークフローネットとなる。

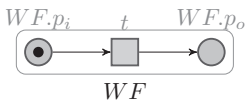
この証明は、ワークフローネットの構造に関する帰納法で容易に示すことができるため、省略する。

注意 3.1 それぞれのワークフローが共通のプレース、トランジションを持つ場合、構成されたペトリネットが再びワークフローになるかは自明ではない。特に、「参照」を経由し共有されたトランジションやワークフローがある場合には注意が必要である。

以下、それぞれの直観的意味、図的表示、形式的意味を与える。

3.5.1 トランジションラッパー (Transition Wrapper)

トランジションは、開始プレース (p_i) 、終了プレース (p_o) をつけ加えることでワークフローネットとすることができる。



$$
\begin{aligned}
WF &::= t \in T \\
WF.P &= \{WF.p_i, WF.p_o\} \\
WF.T &= \{t\} \\
WF.F &= \{(WF.p_i, t), (t, WF.p_o)\}
\end{aligned}
$$

図 2: 「トランジションラッパー」の図的表示とその形式的意味

3.5.2 ワークフローネット参照

ワークフローネット参照により、他のワークフローネットを現在のコンテキストにおいて使用することができる。このため、外延的な意味は参照先のワークフローネットと等しい。

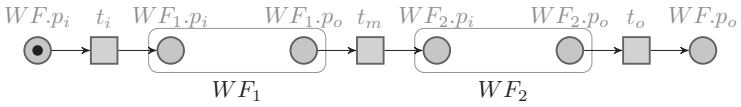
$$
\begin{aligned}
WF &::= \textbf{reference of } WF_1 \\
WF.P &= WF_1.P \\
WF.T &= WF_1.T \\
WF.F &= WF_1.F
\end{aligned}
$$

図 3: 「参照」の形式的意味

3.5.3 逐次結合

ワークフローネットの逐次結合とは、複数のワークフローネットを直列に接続して構成するワークフローネットである。$WF_1 \to WF_2$ の場合、WF_1 の終了プレースと WF_2 の開始プレースを決

定的 (内部) トランジションを介して接続する。$WF_1 \to WF_2$ のとき、WF_1 を WF_2 のプリフィックス、もしくはガードと呼ぶ。

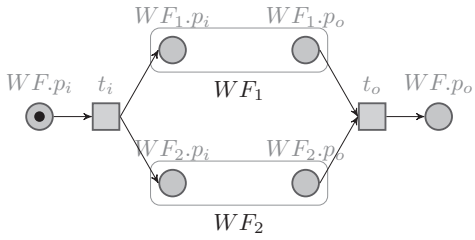


$$
\begin{aligned}
WF &::= WF_1 \to WF_2 \\
WF.P &= \{WF.p_i, WF.p_o\} \cup WF_1.P \cup WF_2.P \\
WF.T &= \{t_i, t_m, t_o\} \cup WF_1.T \cup WF_2.T \\
WF.F &= \{(WF.p_i, t_i), (t_i, WF_1.p_i), (WF_1.p_o, t_m), (t_m, WF_2.p_i), (WF_2.p_o, t_o), (t_o, WF.p_o)\} \\
&\quad \cup WF_1.F \cup WF_2.F
\end{aligned}
$$

図 4: 「逐次結合」の図的表示とその形式的意味

3.5.4 並列結合

ワークフローネットの並列結合とは、複数のワークフローネットを並列に接続して構成するワークフローネットである。構成子のワークフローネットはすべて発火可能となり、全体のワークフローネットはそれを構築するすべてのワークフローネットの終了をもって終了する。

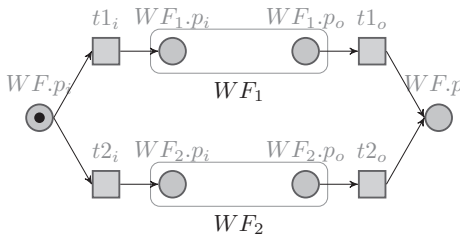


$$
\begin{aligned}
WF &::= WF_1 \otimes WF_2 \\
WF.P &= \{WF.p_i, WF.p_o\} \cup WF_1.P \cup WF_2.P \\
WF.T &= \{t_i, t_o\} \cup WF_1.T \cup WF_2.T \\
WF.F &= \{(WF.p_i, t_i), (t_i, WF_1.p_i), (WF_1.p_o, t_o), \\
&\quad (t_i, WF_2.p_i), (WF_2.p_o, t_o), (t_o, WF.p_o)\} \\
&\quad \cup WF_1.F \cup WF_2.F
\end{aligned}
$$

図 5: 「並列結合」の図的表示とその形式的意味

3.5.5 選択結合

ワークフローネットの選択結合とは、複数の構成子ワークフローネットのうちただ一つ選択するためのワークフローネットである。構成子のワークフローネットはすべて発火可能となるが、実際に選択されるワークフローネットはそのうちの一つである。構成子ワークフローネットのプリフィックスのうち、一つでも決定性トランジションが混入した場合、他のワークフローネットが選択されることはない。意図的にこの動作を避けるためには、外界とのインタラクションをともなうガードトランジションを使う必要がある。



$$
\begin{aligned}
WF &::= WF_1 \oplus WF_2 \\
WF.P &= \{WF.p_i, WF.p_o\} \cup WF_1.P \cup WF_2.P \\
WF.T &= \{t1_i, t1_o, t2_i, t2_o\} \cup WF_1.T \cup WF_2.T \\
WF.F &= \{(WF.p_i, t1_i), (t1_i, WF_1.p_i), \\
&\quad (WF.p_i, t2_i), (t2_i, WF_2.p_i), \\
&\quad (WF_1.p_o, t1_o), (t1_o, WF.p_o), \\
&\quad (WF_2.p_o, t2_o), (t2_o, WF.p_o)\} \\
&\quad \cup WF_1.F \cup WF_2.F
\end{aligned}
$$

図 6: 「選択結合」の図的表示とその形式的意味

4 具現化

本章ではこれまでのペトリネットとワークフローネットの定義に基づく e-Learning コース生成システムの具現化について説明する。

本システムは、主に以下の部分からなる。

- ワークフローネットの定義
- ワークフローネットのペトリネットへの変換
- ペトリネット実行エンジン
- 問題画面定義

まず、コース設計者は XML 形式でワークフローネットの定義をおこない、変換機により JavaScript 言語形式のペトリネットに変換する。変換機は python とその標準クラスライブラリで記述されており、変換作業はごく短時間 (コースのプレースとトランジションの総数の和に対して線形時間) に、かつ、自動的に行うことができる。変換後の JavaScript は、同じく JavaScript のペトリネット実行エンジンによって解釈される。画面設計は HTML5 によるいわゆる Web ページとして記述し、ワークフローネット、ペトリネット実行エンジンとあわせて学習教材を構成する。

4.1 ワークフローネットの定義

ここでは、トランジションの種類とその具体的な機能、および、ワークフローネット構築子を XML 形式で定義する方法について述べる。

4.1.1 トランジション

トランジションには大きく分けて「通常のトランジション」と「ガード (guard) トランジション」の 2 種類がある。ガードトランジションとは、非決定性を持つべき選択合成時での決定的遷移を防ぐために利用される「ガード」である。通常のトランジションは、解釈と同時にフロー関係 $PN.F$ に組み込まれるのに対し、ガードトランジションはフロー関係への導入がいったん猶予される。当

該トランジションを要求するワークフローネットが解釈された時にそのワークフローネットのプリフィックスとして逐次合成され、フロー関係に組み込まれる。

次に、トランジションの機能について説明する。発火可能なトランジションはイベントを契機に発火する。イベントには e、message、emphasis、normalize、hide、show イベントなど自発的 (決定的) に発火するものと、click イベント (ユーザからのボタンクリックに対応) 、change イベント (ユーザからのキー入力に対応) などの、外部とのインタラクションを伴うものがある。ユーザからの入力を得たり、ユーザの画面に変化をもたらすトランジションには wid (Web object ID) を指定が必須である。トランジションに対応するペトリネットは図 7,8 に、トランジションとその機能については表 1 に掲載する。

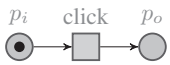


図 7: ボタンクリック (click)

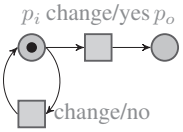


図 8: ユーザ入力 (change)

Tr イベント	アクション	必須の属性	機能
e	FireNow		すぐに発火
click	clicked	wid	ボタンクリックに対応
change	changed	wid, answer	入力と答えを照合
message	message	wid, message	Dom オブジェクトに「message」を表示
emphasis	emphasis	wid	Dom オブジェクトのテキストを強調
normalize	normalize	wid	Dom オブジェクトのテキストの効果をなくす
hide	hide	wid	Dom オブジェクトを隠す
show	show	wid	Dom オブジェクトを表示する

表 1: トランジション/イベント/アクションの対応

4.1.2 ワークフローネット

ワークフローネットを定義する XML ファイルの記述は非常にシンプルである。トップレベルの DOM 要素として唯一の `<WorkFlowPetriNets>` をもち、その子孫として複数のワークフローネット (コース) を登録する。次の例では、並列結合型 (type=”Parallel”) のワークフローネット `Q1_1`、逐次結合型 (type=”Sequential”) のワークフローネット `Q1_2`、選択結合型 (type=”Selective”) のワークフローネット `Q1_3` の 3 つのワークフローネットを定義している。

XML によるワークフローネットの記述例

```
<?xml version="1.0"?>
<WorkFlowPetriNets>
  <WF name="Q1_1" type="Parallel"/>
  <WF name="Q1_2" type="Sequential"/>
  <WF name="Q1_3" type="Selective"/>
</WorkFlowPetriNets>
```

4.1.3 トランジションラッパー

トランジションはそれ自身トップレベルのワークフローネットとなることはできず、他のワークフローネットの子要素として記述される。この時、トランジションは自動的にラップされ、ワークフローネットの子要素として登録される。次の例では、逐次的ワークフローネットの子要素としてトランジションを並べて記述することで、WTC1、WTC2、WTC3 の Web オブジェクト ID をもつボタンを順にクリックするワークフローネットを生成する。

トランジションラッパーの例

```
<WF name="Q1" type="Sequential">
  <Transition event="click" name = "TC1" wid = "WTC1"/>
  <Transition event="click" name = "TC2" wid = "WTC2"/>
  <Transition event="click" name = "TC3" wid = "WTC3"/>
</WF>
```

4.1.4 逐次結合 (Sequential)

逐次結合は子要素を逐次的 (連続的) に発火可能にするワークフローを定義する。Sequential 構築子は 0 個以上の子要素をもつ。

逐次結合の記述例

```
<WF name="Q1" type="Sequential">
  <WF name="Q1_1"/>
  <WF name="Q1_2"/>
  <WF name="Q1_3"/>
</WF>
```

この例では 3 つの子要素を持つ逐次的ワークフローネットを定義しており、次のような形式的意味をもつ。

$$WF_{\texttt{Q1}} ::= WF_{\texttt{Q1\_1}} \rightarrow (WF_{\texttt{Q1\_2}} \rightarrow WF_{\texttt{Q1\_3}})$$

4.1.5 並列結合 (Parallel)

並列結合は子要素をすべてを同時に発火可能にするワークフローネットを定義する。Parallel 構築子は 1 個以上の子要素をもつことができる。全体のワークフローネットが終了するためには、子要素すべての終了が必須である。

並列結合の記述例

```
<WF name="Q1" type="Parallel">
  <WF name="Q1_1"/>
  <WF name="Q1_2"/>
  ...
  <WF name="Q1_n"/>
</WF>
```

この例のワークフローネットの形式的意味は次のようになる。

$$WF_{\texttt{Q1}} ::= \bigotimes_{\texttt{i}\in\{1,\ldots,n\}} WF_{\texttt{Q1\_i}}$$

4.1.6 選択結合 (Selective)

選択結合は子要素すべてを同時に発火可能にするワークフローネットを定義する。Selective 構築子は 1 個以上の子要素をもつことができる。ただし、そのうち一つのワークフローネットが発火した場合、他のワークフローネットは発火不能となる可能性がある。

選択結合の記述例

```
<WF name="Q1" type="Selective">
  <WF name="Q1_1"/>
  <WF name="Q1_2"/>
  ...
  <WF name="Q1_n"/>
</WF>
```

この例のワークフローネットの形式的意味は次のようになる。

$$WF_{\mathtt{Q1}} ::= \bigoplus_{\mathtt{i} \in \{1,\ldots,n\}} WF_{\mathtt{Q1\_i}}$$

4.1.7 ワークフローネット参照

いったん定義された (明示的な名前を持つ) ワークフローネットは他の文脈から参照/共有されることができる。ワークフローネットの参照は、単にワークフローネットの名前を指定することで行われる。この時、単に type 宣言を省略するか、type=”Reference” とすればよい。

4.2 ペトリネット実行エンジン

ペトリネット実行エンジンはほぼペトリネット発火規則 (定義 3.4、3.5) どおりに実装している。ただし、本システムで多用されている ”e”（空）トランジションなど、ユーザや外部環境とのインタラクションを必要としない決定的トランジションは、「発火可能になると同時に猶予なく発火する」という規則が暗黙裡に付加されている。

異なるふるまいをもつ選択結合の例

```
<WF name="Q1" type="Selective">
  <Transition event="click" name = "TQ1" wid = "WTQ1"/>
  <Transition event="click" name = "TQ2" wid = "WTQ2"/>
  <Transition event="e"/>
</WF>
<WF name="Q2" type="Selective">
  <Transition event="click" name = "TQ1" wid = "WTQ1"/>
  <Transition event="click" name = "TQ2" wid = "WTQ2"/>
</WF>
```

例のワークフローネット Q1 は、それ自体が発火可能になったと同時に e(空) 遷移がおこなわれるため、TQ1 や TQ2 のクリックイベントが選択されることは決してない。これに対して、ワークフローネット Q2 は TQ1 か TQ2 のどちらか一つを必ず選択する。このように、Q1 と Q2 のふるまい的な性質は全く異なる。

4.3 開発環境

参考までに、本システム開発で使用した環境を記しておく。現時点においてすべてフリーで手に入るものとその標準ライブラリを使用しており、特定の OS に依存しないコードになっている。

システム名	バージョン	用途
Google Chrome	31.0.1650.63 m	Web ブラウザ
Firefox	26.0	Web ブラウザ
jQuery	1.10.2	Javascript 開発
CryptoJS	v3.1.2	答えの暗号化 (SHA256)
Python	2.7.3	コース記述 XML ファイルの処理

表 2: 開発環境

5 コース実装例

ここでは、本システムで「16 進数で書かれた $fe4_{(16)}$ を 8 進数表記にしなさい」という問題の解法を説明するコースの作成例を紹介する。

5.1 コースシナリオ

「16 進数を 8 進数に変換する」という問題は「16 進数を 10 進数に変換する」という問題と、「10 進数を 8 進数表記する」という問題の「逐次合成」であると考えるのが妥当であろう。また「16 進数を 10 進数に変換 (ステップ 1-1,1-2)」したり、「10 進数を 8 進数に変換 (ステップ 2)」するプロセスはそれぞれさらに細かなステップから合成されている。図 9 にコースの概要と、対応するワークフローネット図を示す。なお、紙面の関係上、ワークフローネット図は大幅に簡略している。

- 16 進数を 8 進数に変換する
  - ガイドを見る (オプション)

(ステップ 1) 16 進数を 10 進数に変換する

1. 16 進数の「数字」を 10 進数で読み替える (ステップ 1-1)
2. 16 進数の桁の「重み」を 10 進数で表記する
3. それぞれの桁が表している数を 10 進数で表記する (ステップ 1-2)
4. 10 進数表記したものをすべて合計する

(ステップ 2) 10 進数を 8 進数に変換する

1. 8 進数の桁の重みを 10 進数で表記する
2. 変換する前の数から 8 進数の「おおきな」かたまりがいくつとれるか計算する
3. 元の数字から 8 進数のかたまりを引き算する
4. かたまりがとれなくなるまでを繰り返す
5. とれたかたまりの個数を並べて表記する

(ステップ 3) 結果を記述する

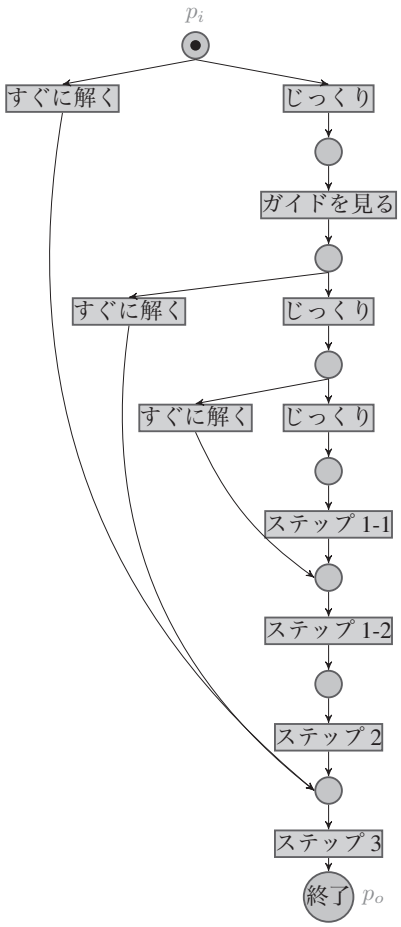


図 9: N 進数変換コース概要とそのワークフローネット

5.2 画面定義

「N 進数変換」コースの HTML5 による画面定義ファイルは次のようになっている。なお、紙面の都合上「ステップ 1」と「ステップ 2」の表示の一部分にとどめている。コースの記述と画面定義を分割することにより、簡潔な記述になっていることがわかる。

「N 進数変換」コースの 画面定義

```
<h1>【200】  16 進数で書かれた fe4<sub>(16)</sub>を 8 進数表記にしなさい</h1>
<div class='menu' id='Menu01'>
  <button id='Menu01_1'>すぐに解く</button>
  <button id='Menu01_2'>ガイドをみる</button></div>
<div id='Guide' hidden>...</div>
...
<div id='Step1' hidden>...</div>
<div id='Step2' hidden>
  <h2>(2) 4068<sub>(10)</sub> を 8 進数に変換する</h2>
  <div class='menu' id='Menu2' hidden>
    <button id='Menu2_1'>すぐに解く</button>
    <button id='Menu2_2'>じっくり解く</button></div>
  <div id='Step2_1' hidden>
    <h3>8 進数の桁の重みを 10 進数で表記する</h3>
    <table class="center">
      <tr>
        <th>8<sup>4</sup></th><th>8<sup>3</sup></th>
        <th>8<sup>2</sup></th><th>8<sup>1</sup></th>
        <th>8<sup>0</sup></th></th></tr>
      <tr>
        <td><input id='Rank_8_4'/></td><td><input id='Rank_8_3'/></td>
        <td><input id='Rank_8_2'/></td><td><input id='Rank_8_1'/></td>
        <td><input id='Rank_8_0'/></td></tr>
    </table></div>
  <div id='Step2_2' hidden>
  <h3>元の数字から 8 進数のかたまりを引き算する</h3>
    <table class="center">
      <tr id='Step2_2_1' hidden>
        <td align="left">  4068 から 8<sup>4</sup> (= 4096) が</td>
        <td> <input id='Quotient_8_4'/>個とれ、そのあまりは
             <input id='Remnant_8_4'/></td></tr>
      <tr id='Step2_2_2' hidden>
        <td align="left">  4068 から 8<sup>3</sup> (= 512) が</td>
        <td> <input id='Quotient_8_3'/>個とれ、そのあまりは
             <input id='Remnant_8_3'/></td></tr>
      <tr id='Step2_2_3' hidden>
        <td align="left">  484 から 8<sup>2</sup> (= 64) が</td>
        <td> <input id='Quotient_8_2'/>個とれ、そのあまりは
             <input id='Remnant_8_2'/></td></tr>
      <tr id='Step2_2_4' hidden>
        <td>  36 から 8<sup>1</sup> (= 8) が</td>
        <td> <input id='Quotient_8_1'/> 個とれ、そのあまりは
             <input id='Remnant_8_1'/></td></tr>
      <tr id='Step2_2_5' hidden>
        <td>  4 から 8<sup>0</sup> (= 1) が</td>
        <td> <input id='Quotient_8_0'/> 個とれ、そのあまりは
             <input id='Remnant_8_0'/></td></tr>
      <tr id='Step2_2_6' hidden>
        <th colspan='2'>  4068 <sub>(10)</sub> を 8 進数に変換すると
            <input id='Step2_2_Answer'/><sub>(8)</sub></th></tr>
    </table></div></div>
<div id='Step3' hidden>...</div>
```

5.3 コース実行画面

最後に、コースウェアの実行画面をいくつか紹介する。図 10、11、12 はそれぞれ「コースの細かさを指定するメニュー」、「16 進数を 10 進数に変換するコース」、「10 進数を 8 進数に変換するコース」のスクリーンダンプイメージである。

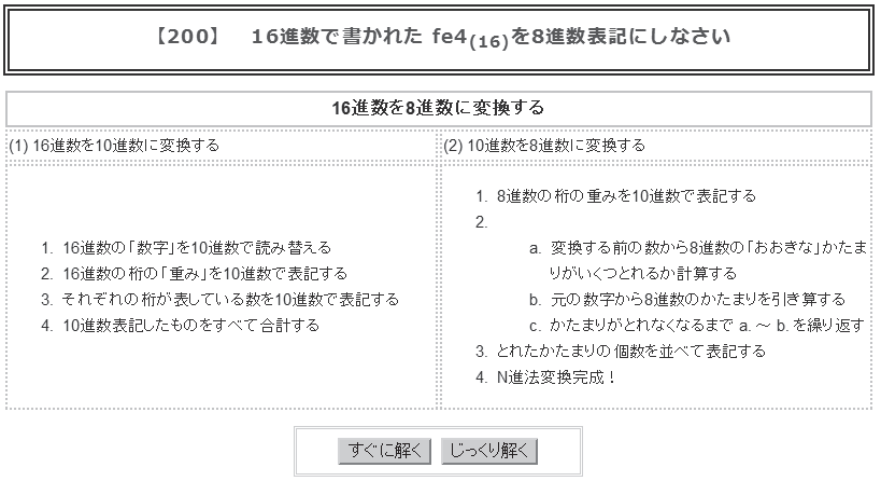


図 10: ガイド表示画面

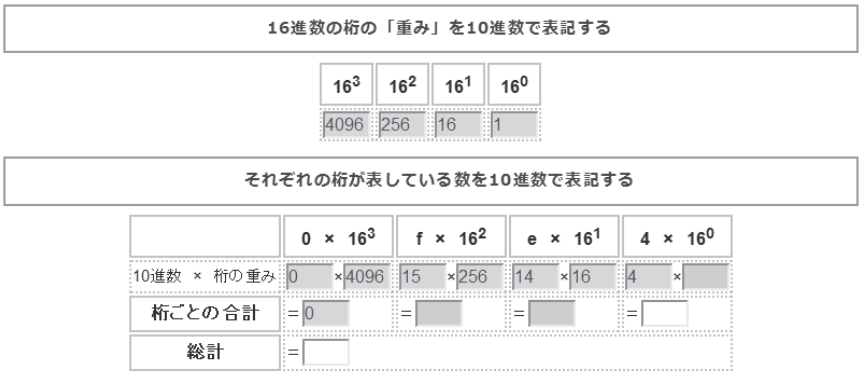


図 11: 16 進数 → 10 進数変換

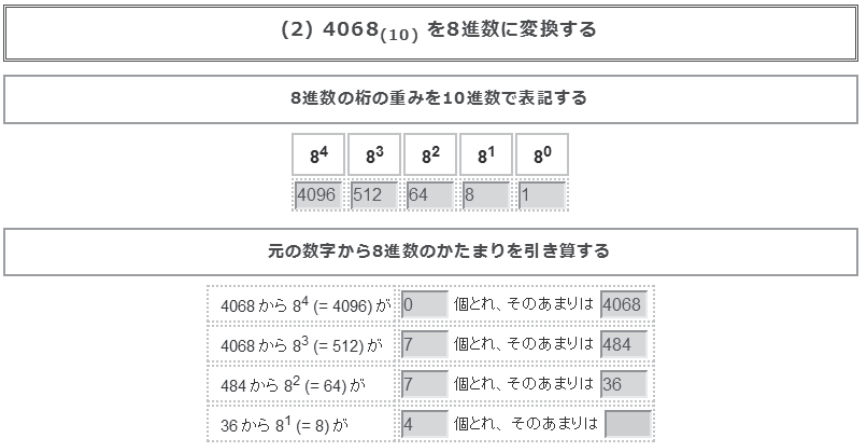


図 12: 10 進数 → 8 進数変換

6 おわりに

本論文では、リメディアル教育にみられる学生の理解度のばらつきに対応するため、キメ細かな e-Learning コースの必要性を主張した。そして、キメ細かなコースをペトリネットのサブクラスであるワークフローネットを記述するための XML 形式の簡易記述言語を定義した。XML 形式のコースは自動的に JavaScript のペトリネットに変換され、実行エンジンと組み合わせて実行される。この仕組みにより、コース作成工程からプログラミングを取り除くことができる。すなわち、教材作成者は「XML によるコース定義」と「HTML5 による画面定義」の 2 つの工程に専念することができる。同時に、コースの定義、画面定義工程を分離することによりそれぞれの工程が非常に簡潔なものとなることを示すことができた。

6.1 今後の課題

本研究は、ワークフローネットの技術を e-Learning コースウェアに応用するというアイデアを示し、実装することでその実現可能性を示したが、実際の授業に用いた経験はない。実用に耐えうるシステムにしていくための残された課題を列挙し本論文の締めくくりとする。

- 経験の蓄積: リメディアル教育に必要な練習問題を洗い出し、本コースにとりこみ、実際に授業を行い効果を確認する必要がある。
- オブジェクト化：他のコースの教材を継承し、カスタマイズして再利用する仕組みが必要である。
- 問題とコースとの分離: 反復練習を支援するために、たとえば、問題の数値だけ変更して同様の問題を量産することが求められる。
- LMS の実装:実用的な e-Learning システムには LMS は必須の機能である。

- カラーペトリネットへの拡張:本研究では単独の学習者を想定しいているため、単色のトークンからなるペトリネットを用いた。複数の学習者を支援するためには、学習者が異なることなどを表す「色付きのトークン (coloured token)」を使用できるペトリネットに拡張していかなければならない。
- 統計的解析:コースが学習者にとって使い勝手が良いものであるかを客観的に判断するためには、学習者の成績や作業時間などの統計的な解析が必要である。
- 最適化の仕組み:本論文のワークフローネット構築子による素直な実装は、不要のプレースやトランジションを大量に生成する。これは、次の分析/検証作業をいたずらに複雑にするため、プレースやトランジションの数に関する最適化の仕組みが必要である。
- 分析/検証: 本文中でも指摘があるが、ワークフローネットをワークフローネット構築子を用いて合成した結果は必ずしもワークフローネットになるとは限らない。これに限らず、ペトリネットの健全性、安全性、活性などの保証はコースの重要な性能の一つである。
- 柔軟な画面設計:タブレット端末、スマートフォンへの対応: PC の広い画面での学習だけでなく、コンパクトな画面での学習を支援しなくてはならない。
- ユーザフレンドリーな開発環境: 本システムはプロトタイプの段階にあり、コースの記述の不具合は XML Parser や JavaScript などのエラーメッセージを手掛かりにデバッグしている。Web プログラミングの知識の乏しいユーザにも使いこなせる開発環境にする必要がある。

参考文献

[1] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE, Vol. 77, No. 4. IEEE Computer Society*, pp. 541–580, April 1989.

[2] van der Aalst, Wil M. P. The application of petri nets to workflow management. *The Journal of Circuits, Systems and Computers*, Vol. 8, No. 1, p. 2166, 1998.

[3] H. M. W. Verbeek. Analyzing bpel processes using petri nets. In *Florida International University*, pp. 59–78, 2005.

[4] 青木久美子. e ラーニングの理論と実践 (放送大学大学院教材). 放送大学教育振興会, 2012.

[5] 辻 孝吉太田 淳. ネット理論 –ペトリネットとその解析問題–. *IEICE ESS Fundamentals Review, Vol. 2 (2009) No. 4*, pp. 56–67, May 2009.

[6] 村田忠夫. ペトリネットの解析と応用. 近代科学社, 1992.

〔抄　録〕

本論文では、リメディアル教育にみられる学生の理解度のばらつきに対応するため、キメ細かな e-Learning コースの必要性を主張した。

さらに、キメ細かなコースをペトリネットのサブクラスであるワークフローネットを記述するための XML 形式の簡易記述言語を定義した。XML 形式のコースは自動的に JavaScript のペトリネットに変換され、実行エンジンと組み合わせて実行される。この仕組みにより、コース作成工程からプログラミングを取り除くことができる。この仕組みにより、教材作成者は「XML によるコース定義」と「HTML5による画面定義」の２つの工程に専念することができる。加えて、コースの定義、画面定義工程を分離することによりそれぞれの工程が非常に簡潔なものとなることを示すことができた。