RPA のための実行前検査に関する研究

2018SE087 鈴置真輝
指導教員：張漢明

1　はじめに

近年，DX(Digital Transformation) の推進により，アプリケーション業務の効率化が進められている．RPA(Robotic Process Automation) はアプリケーションの操作をソフトウェアで自動化する技術であり，ワークフローをノーコードで作成することにより，作業の自動化を目指している．RPA を DX の枠組みで利用するためには，並行性を含んだ複数のアプリケーションとの連携，および，ワークフローが法令や業務内の規則などが守られていることを保証する必要がある．ワークフローを検証するためには，ワークフローの仕様を記述して，ワークフローがその仕様を満たしていることを検証する必要がある．

本研究の目的は，CSP[1] を用いた RPA のワークフローの検証を支援することである．CSP は，並行システムを記述・検証するための理論である．CSP では，詳細化関係を用いて仕様と実現の間の関係を検証することができる．ワークフローの連携とワークフローの仕様を記述するための CSP 記述を提供することにより，RPA のワークフローの検証を支援することができると考えた．

本研究の研究課題は，

1.「ワークフローの連携」と「仕様」の定式化

2.「ワークフローの連携」と「仕様」の表記法

を提案することである．本研究では, 以下のようなアプローチで研究を行った．

1. CSP で検証するためのワークフローの基本構造を特定
2. 基本構造を CSP の関数で定式化

2　既存研究と関連技術

2.1　DX における RPA の位置づけ

RPA はアプリケーションの操作をソフトウェアで自動化することにより, 人間が行う業務の効率化を目指している．DX は, 現場主導 DX と, システム基幹 DX に分けられており,RPA は現場主導 DX に属している [2]. 現場主導 DX は, 頻度は少ないが, 要件は多岐にわたる業務のことを指し, システム基幹 DX は, 要件が明確で量が多い業務のことを指す．大石によると, この先, 現場主導 DX と, システム基幹 DX の連携が必須になると述べている．

2.2　CSP の概要

CSP では, アクションや操作などをイベントとして抽象化し, イベントを用いて対象システムの振る舞いを記述する.CSP では, 接頭辞, 再帰, 選択を用いて逐次プロセスを構成し, 並行性はプロセスを合成することで表現する．

逐次プロセス

逐次プロセスは, 接頭辞, 再帰, 選択から構成される．

- 接頭辞：`A = x -> P`

プロセス A は，イベント x の後で，プロセス P として振る舞うことを表している．

- 再帰：`CLOCK = tick -> CLOCK`

プロセス CLOCK は，イベント tick を繰り返す時計の振る舞いを表している．

- 選択：`B = x -> P | y -> Q`

プロセス B は，イベント x の後で P として振る舞うか，もしくは，イベント y の後で Q として振る舞うことを表している．

並行プロセス

並行プロセスとは同時並行して実行されているプロセスの集合である．

```
channel coin,choc,clink,clunk,toffee,curse

NOISYVM = coin -> clink -> choc -> clunk -> NOISYVM

CUST = (coin -> (toffee -> CUST []
curse -> choc -> CUST))

X2 = NOISYVM [{coin,choc,clink,clunk,toffee}||
{coin,choc,toffee,curse}] CUST
```

プロセス X2 はプロセス NOISYVM, プロセス CUST を並行合成したプロセスである.NOISYVM と CUST で共通のイベントがある場合に限り, 同期をとる．

詳細化関係

トレース詳細化, 失敗詳細化, 失敗発散詳細化の 3 つが存在する. トレース詳細化 ($T \subseteq_T SPEC$) は安全性検証に用いることができ, 失敗詳細化 ($T \subseteq_F SPEC$) はデッドロックフリーの検証, 失敗発散詳細化 ($T \subseteq_{FD} SPEC$) はライブロックフリーの検証に利用することができる．

3　検証手法の提案

3.1　検証の概要

CSP を用いて, ワークフローを検証するための基本的なアイデアを以下に示す．

- 1 つのワークフローを逐次プロセスで表現
- 連携は, 並行プロセス通信の同期を利用
- 仕様を CSP で表現
- 隠蔽と詳細化関係を用いてプロセスの等価性を検査

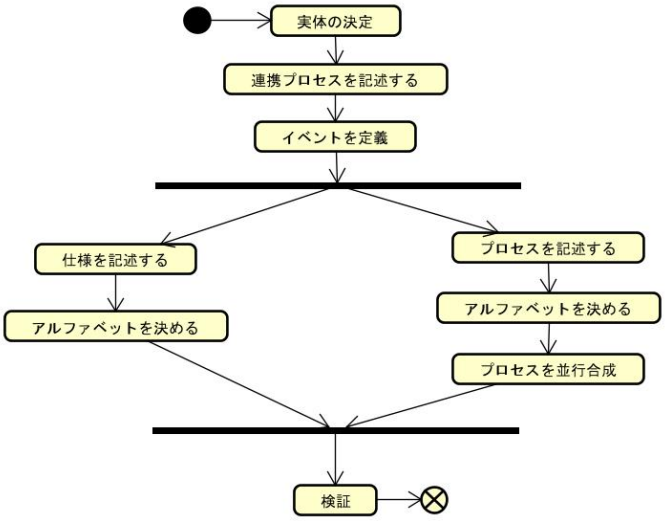

図 1 ガイドラインのアクティビティ図

検証のガイドラインをアクティビティ図を用いて図 1 に示す．アクティビティ図と上で述べたアイデアを対応した CSP 記述を一部事例を用いながら以下に示す.

並行に関する事例として, 複数人でのスケジュール調整の問題を取り上げる．流れとして, 作成者が複数の日時を回答者に提案する. 回答者は自分の都合のつく日時をメールで回答し, 送信する. 作成者は全員の返信を待った後, 確定した日時を全員にメールで報告する. というものである.

1. `channel mail:Attendee.Contents`
   このように記述することで Attendee と Contents というデータを乗せた mail というチャネルを定義でき, 同期を用いるためのデータを定義することができる.
2. `SR(n) = mail.n!reqSchedule -> mail.n?x -> SKIP`
   `Att(n) = mail.n?x -> mail.n!schedule -> SKIP`
   このように記述することで, 連携を並行プロセスの同期を用いて表現することができる.SR を作成者,Att を参加者とし, 上記で定義した mail を CSP の通信を用いて表現した.
3. `SCH = deciAnswer -> SRs({1..N})`
   `ATT = Atts({1..N})`
   SCH を作成者,ATT を参加者の振る舞いとし, それぞれをプロセスとして記述した.
4. `FLOW = SCH [|{|mail|}|] ATT`
   作成者と参加者のプロセスを合成し, 新たな 1 つのプロセスとして定義した.
5. `assert SPEC [T= FLOWY`
   あらかじめ定義したワークフローの仕様である SPEC プロセスと先ほど定義した FLOW プロセスを用いて検証を行う．FDR[3] の assert を用いることで, 検証することができ, トレース詳細化を用いることで研究することができる.

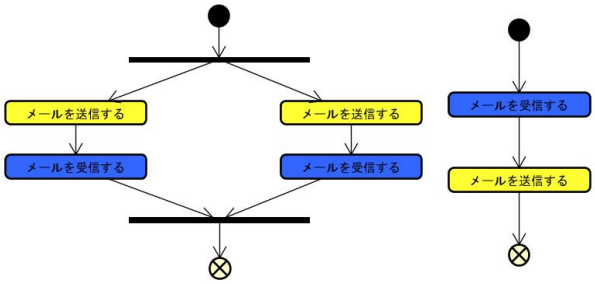

図 2 色付け表記法の例

トレース詳細化を用いて,SPEC と FLOWY のアルファベットが等しいかどうか検証する. 等しかった場合,TRUE が返され, 等しくなかった場合,FALSE が返される. 検証結果,TRUE が返され, このワークフローは仕様を必ず満たすことが保証された.

4 考察

4.1 ガイドラインの有効性

ガイドラインの有効性について評価する. 事例について, ガイドラインに沿うことで検証を行うことができた. また,「メール送信時, 暗号化を行う」や,「参加者が各々スケジュールの空を確認する」といったアクションを追加した, より複雑な事例に対しても同様に検証することができたので, このガイドラインは有効性があるといえる.

4.2 UML を用いた表記法

ワークフローで考慮しなければならない部分として, 同期を取る部分をどのように UML で表記するかという点である. 解決策の 1 つとして, 同期をとるアクションに色を付けることを考えた. このようにすることで, 一目で同期をとるとわかることができ, 全体の把握がしやすくなると考えた. 図 2 に同期法の例を示す.

5 おわりに

本研究では,RPA のワークフローの検証を支援するために,CSP を用いてガイドラインを提示し, 検証を行った. 今後の課題として,RPA,CSP の有識者がお互いの技術を使えるようにするため,CSP と UML の互換性を高めるための方法や,1 対多通信だけでなく, 他対多の通信の記述法, より複雑な, 複数の仕様について対応できるガイドラインを提示することなどが挙げられる.

参考文献

[1] C. A. R. Hoare: Communicating Sequential Processes, Prentice-Hall, 1985.

[2] 大石晴夫：現場手動 DX を実現する業務ナビゲーション技術，電子情報通信学会誌，Vol. 104, No. 8, 2021.

[3] FDR4：https://cocotec.io/fdr/.