社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－ＳＥ－160 (14)
2008－EMB－ 9 (14)
2008／6／20

動作要求のモデル化と管理

神田 未来, 藤倉 俊幸

イーソル株式会社リサーチ＆コンサルテーションサービス部

アブストラクト

動作に関する要求を，マインドマップによりブレークダウンしさらにプロセス代数式(FSP)によって形式仕様化する手法を提案する．提案手法では，形式化した仕様を組み合わせることで動作モデルを生成することができる．この際，仕様間に矛盾が無いことを検証することができる．動作モデルは，タスク設計情報として利用できる．

The Modeling and Managing of the Behavioral Specification

Miki KANDA, Toshiyuki FUJIKURA

Research & Consultation Services Group, eSOL Co., Ltd.

Abstract

This paper focuses attention on behavior requirements, and extracts behavioral specifications by using the MindMap from them. The specifications are formalized to FSP descriptions. The composition of FSP can make a conflict-free behavior model by the model checking. The behavior models are available for the designing of the task structure for the multitasking system.

1. はじめに

実世界の事物を制御対象とする組込みシステムでは，動作に対する要求を矛盾無く定義する事が必要である．従来，要求工程成果物に対する検証はレビューを中心としており，工数が掛かると共に客観性に欠ける等の品質上の問題がある．形式仕様記述やモデル検査を導入する試みがおこなわれているが，述語論理や時相論理などの知識を必要とし多くの一般的開発現場では扱いづらくまだ十分に使いこなせているとは言えない．また，組込みソフトの様な物を動かすための仕様を論理式で効率よく記述できるか不明な点もある．本研究では，要求から動作要求を抽出し直接 FSP(Finite State Process)式[1]で形式化することを検討した．

動作に関する要求は，イベントや動作，状態の生起順序または禁止として仕様化できる．さらにその動作仕様を FSP 式によって形式化した．FSP 式はプロセス代数式を簡略化したものである．FSP 式にすることで，矛盾する動作仕様は合成した際にデッドロックとして検出することができる．ここで，合成とは FSP 式をモデル検査ツールに入力し並列合成することを指す．自然言語から FSP 式への変換は予め用意した変換パターンを使用しておこなったため特に予備知識を必要としなかった．モデル化および検証は LTSA[1]を使用した．LTSA は，FSP を合成し LTS(Label Transition System)に変換しグラフィ

カルに表示することができる. また, LTS に従ったシミュレーションや LTL(Linear Temporal Logic)で表現した性質に対するモデル検査をおこなうことができる.

サンプルとしてカレーの作り方を対象とした. 料理の作り方はほとんど手順の記述であり動作要求のサンプルとして有効である.

「地球に優しい」あるいは「健康によい」等の要求を, カレーを作るための具体的な手順に落として動作仕様とした. 複数の動作仕様が生成され, それらを選択して要求別のカレーの作り方を動作モデルとして生成することができた. 生成する際に矛盾するオプションを指定するとデッドロックとなる. このことで, 実行可能なオプションセットであることを検証できる.

本研究で試行した手法は, 組込みシステム開発時には, 要求工程で利用することで動作要求の抽出と検証およびトレーサビリティの管理を支援できる. この手法の成果物は, 動作仕様集合と動作モデルである. 動作仕様集合から要求に応じて動作仕様を選ぶことで容器有を満足する動作モデルを生成することができる. このことは製品系列を設計する際の可変性の管理に応用することが可能である. また生成される動作モデルは, システムの振る舞い仕様としてタスク設計への入力情報として使うことができる.

2. 要求の仕様化

要求の粒度を揃え, 具体化し, 検証可能にすることを, ここでは仕様化すると呼ぶ. 粒度が大きすぎると, 単なる方針やものの見方に関する意見に過ぎず, 具体的な動作に結びつかない. 「地球にやさしいカレーを作ろう」と大きなこと言っても具体性が無いため実現できるか検証するには至らない. 「地球にやさしい」→「水を大切にする」→「少ない水で洗う」の様にブレークダウンすることで, 「地球にやさしい」ことと「カレーを作ること」の関連が見えてくる. さらに, タマネギを洗うか洗わないかと言う具体的な動作に落とすことで, 動作の矛盾として検出できる. 要求から仕様に落とすことで検証が可能になる(図 1).

本研究では, マインドマップ[1][2]を使用して大きな要求を動作レベルまでブレークダウンした. ブレークダウン過程を示すマインドマップ自

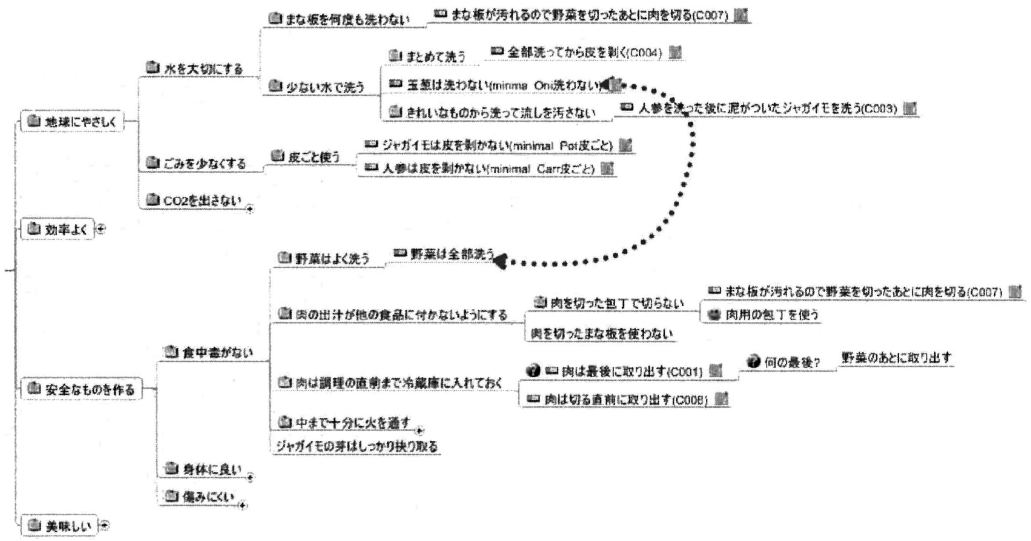


図 1. 要求の仕様化

[1] マインドマップは英国 Buzan Organisation Ltd.の登録商標

身がトレーサビリティ情報になる．また，製品系列を設計する際のオプション構造を示すことになる．上位のパッケージを選択して，動作上の矛盾が発生した場合は，下位の詳細化されたパッケージを使ってオプションを選択し直すことで矛盾を回避した動作を再構成することができる．

動作レベルまでブレークダウンした仕様を，ここでは動作仕様と呼ぶ．動作仕様は次章に述べる方法で FSP 式に形式化する．形式化することで，ある動作をするかしないかと言う直接的な矛盾以外の実行順序に関する矛盾も検出できる様になる．

別の要求から同一の動作仕様が出てくることがある．また一つの要求から互いに矛盾する動作仕様が出てくることもある．これらの要求と仕様の関係を管理するためにマインドマップと要求管理ツールを併用することを現在検討している．動作仕様間の関係としてカレーサンプルでは，矛盾関係と同時に使用する関係の二つが現れた．同時に使用する関係は，方向性がある．これらの仕様間の関係もマインドマップでは無く要求管理ツールで管理・表現する方が効率的であると思われる．

3. 動作仕様の形式化

具体的な動作やイベントにブレークダウンした動作仕様は，動作の実行列として FSP 式で表現する．たとえば，カレーの材料を切るまでの動作は次のように表現する．

P = (_取り出す→_洗う→_皮を剥く→_切る→END).
P 肉 = (_取り出す→_切る→END).

材料を全て切るまでの動作は

||P 切る = (pot:P||carr:P||oni:P||met:P 肉).

の様にして個々の材料に対する切るまでの動作を並列合成することで得ることができる．ジャガイモ(pot)，人参(carr)，玉葱(oni)の基本的動作は共通なので同一のプロセス P をインスタンス化して使用した．pot:P は LTSA におけるインスタンス化表現である．この様にすると，ジャガイモを切る動作は，pot._切ると表現される．動作表現を合わせるために肉(met)についても met:P 肉とした．

個々の動作仕様は，基本動作に追加することを想定して形式化した．たとえば，肉を最後に取り出すと言う動作仕様は，

C001_1 = (pot._取り出す→_野菜取り出した→END).
C001_2 = (carr._取り出す→_野菜取り出した→END).
C001_3 = (oni._取り出す→_野菜取り出した→END).
C001_4 = (_野菜取り出した→met._取り出す→END).
minimal ||C001 = (C001_1 || C001_2 || C001_3 || C001_4) ￥ {_野菜取り出した}.

の様に形式化した．C001 と P 切るを並列合成することで，肉を最後に取り出す条件付きの切るまでの動作を生成する．

以下，個々の動作を a, b,...の様に小文字のアルファベットで表す．

3.1. AND 動作

a と b の動作終了後次に進む動作仕様は以下のように FSP 式で記述できる．

A = (a-→END).
B = (b→END).
||AND = (A||B).

3.2. 排他的 OR 動作

a または b のどちらか一方のみの動作終了後次に進む動作仕様は以下のように FSP 式で記述できる．この時，選択されなかった動作は禁止される．

EXOR = ({a, b}→END).
||ExOr = (EXOR).

この他に，複数の中からどれか二つを実行すると言う動作仕様もパターン化した．

3.3. 禁止

オプションとして基本動作に追加した動作 a

を禁止する FSP 式は，モデリングに使用した LTSA では

NOT = END + {a}.

となる．AND と NOT を合成した場合はデッドロックするが，EXOR と NOT を合成した場合はデッドロックしない．

基本動作に含まれる動作 a を禁止する場合は，a を内部動作として隠蔽することで対応した．LTSA では以下のように表現する．

minimal ‖P 皮ごと = (P)　¥{_皮を剥く}.

3.4. OR 動作

どちらか一方のみを実行する排他的 OR ではなく，両方を実行しても良い含意的論理和(inclusive OR)と同じ OR 動作の場合は，色々な意味があり一つのパターンにすることが困難である．たとえば，a または b の後に c を実行すると言った場合，c を最初に実行しないと言う意味に解釈すると以下のような動作系列を含む．

1. a→c→b
2. b→c→a
3. a→b→c

しかし，実行系列 3 を排除する意図で a または b の後に c を実行すると言う場合もある．たとえば，「豚肉か鶏肉を使う」と言った場合には，一般的には両方は入れないと解釈できる．

また，次の動作 c を含めないと意図が明確にならない等，独立な動作仕様として扱いづらいこともパターン化を困難にする理由である．

a, b, c の中で c を最初に実行しないと言うパターンは以下のように FSP 化できる．

OR = Q0,
Q0　= (b→Q5 | a→Q7),
Q1　= (b→Q2 |a→Q4),
Q2　= (a→Q3),
Q3　= END,
Q4　= (b→Q3),
Q5　= (c→Q2 | a→Q6),
Q6　= (c→Q3),
Q7　= (c→Q4 | b→Q6).

しかし，これと Not a を合成すると，正しく b→c が生成されるが OR プロセスが最後まで実行されないのでデッドロックと判定されてしまい検証結果の解釈に混乱が生じる．このため解釈においても個別の扱いが必要になる．

3.5. 並列動作の記述

FSP で扱う各イベントの起生は，幅を持たずに瞬間的な動作として扱う．広がりを持ち，時間の掛かる動作をモデル化する場合は，その動作の開始と終了を別のイベントとして扱うことでモデル化した．たとえば，幅を持った二つの動作 a, b を順に行う場合は，開始を st,終了を fn として

BeforeAB = (a.st→a.fn→b.st→b.fn→END).

によって形式化した．炊飯等の時間の掛かる動作の並列実行を表現するために必要である．これを，論理式で表現しようとすると

$$\Box(p_a \rightarrow (\neg p_b \wedge Fp_b)) \wedge \Diamond(\neg p_a \wedge \neg p_b \wedge Pp_a \wedge Fp_b)$$

の様な複雑な式[3]になってしまい現場には全く馴染まない．ここで，p_a と p_b は a, b を実行中であることを示す命題，P は過去オペレータ，F は未来オペレータである[3].

時間的な幅を持った事象の起生関係は Allen によりまとめられている．ここでは，同時イベントは扱わないこととして，Allen の分類[5]の内，before, overlap, durring をパターン化した．

4. 動作モデルの生成と利用

動作モデルは基本動作をモデル化したベースモデルと選択された動作仕様から生成した．サンプルにおけるベースモデルのサイズは，約 3000 状態，約 13000 遷移になった．基本的には動作仕様は動作制約として作用するので生

成される動作モデルサイズは小さくなる. ただし, 電子レンジを使用する等の動作要求の場合は, 新たな動作が追加になるためモデルサイズは大きくなる. ベースモデルの選択と要求仕様の追加・変更に対する制限は基本的にないので, 状態爆発が発生しない限りインクリメンタルにモデルを成長させていくことができる.

ベースモデルと個別に FSP 化した動作仕様を合成することで動作モデルを生成する作業は, ルーチン的であるため自動化することを検討した. 当初, 要求管理ツールで動作モデルを生成することを検討したが, ツール依存を避けるために独立なプログラムとすることを検討している(図 2).

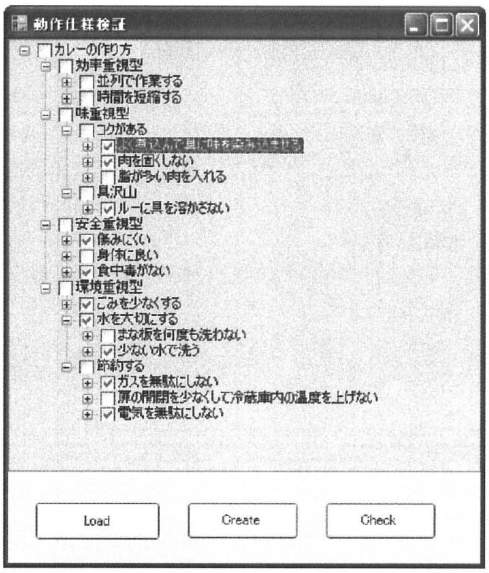


図 2. 動作仕様選択ツール

生成した動作モデルは, LTSA のシミュレータを使うことでカレーを作る際の手順ガイドとして利用できる. どのような手順で作業をすればよいかを示し, 終了した作業にチェックを入れることで次に実施可能な作業を表示してくれる(図 3).

組込みシステム開発では要求仕様から生成した動作モデルは, タスク設計における入力情報として利用できる[6]. カレーモデルに対応させると, たとえばカレー作りを 3 人で分担する場合の作業分担や同期の取り方を決めるための基本情報として利用できる. また, 実際に作成した作業分担の正しさを検証するための検査用モデルとして使用できる.

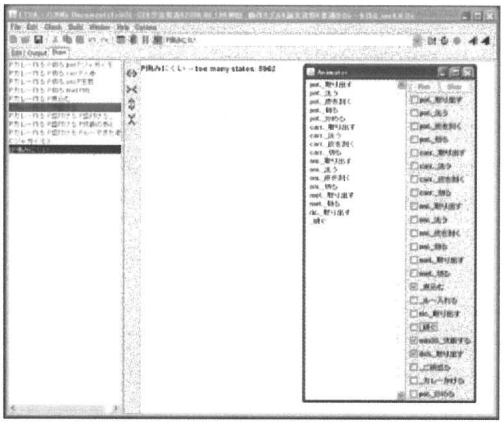


図 3. LTSA によるシミュレーション

5. まとめと今後の課題

組込みシステムで重要な動作要求の仕様化と矛盾の無い仕様の組合せを生成する手法について報告した. 当初, タスク設計情報を得ることを目的としたが, 開発プロセスの設計や作業スケジューリング等にも利用可能ではないかと思われる.

今後は,

1. 要求と仕様の多対多関係および仕様間の矛盾関係や依存関係の管理方法
2. 動作要求以外の仕様化および形式化への拡張
3. ツールを利用した処理の自動化

について検討する.

参考文献

[1] Jeff Magee,Jeff Kramer,Concurrency: State Models & Java Programs,Wiley,2006

[2] Tony Buzan, The Mind Map Book, BB, 1993.
[3] 東条敏, 知の科学 言語・知識・信念の論理, オーム社, 2006.
[4] A. Prior. Past, Present and the Future, Oxford Clarendon Press, 1967.
[5] J. F. Allen, Towards a general theory of action and time, Artificial Intelligence, 23, pp. 123-154, 1984.
[6] 藤倉俊幸, オーバーラップ制御の設計と検証, 第四回システム検証の科学技術シンポジウム, http://unit.aist.go.jp/cvs/symposium/verification2007/abstract2007.html#fujikura