日本ソフトウェア科学会第 27 回大会 (2010 年度) 講演論文集

CAN 製品へのモデルベーステスト適用の取り組み

岡本 圭史　泉 佑治　木下 佳樹　中野 哲　清水 徹

本論文では，CAN 製品のテスト工程へのモデルベーステスト適用に関する，我々の取り組みを紹介する．CAN(Controller Area Network) は，車載ネットワーク用の通信プロトコルとして開発され，現在では，産業機械やファクトリーオートメーションなどの製品でも利用されるなど，その利用範囲は大きい．したがって，CAN 製品には高い信頼性が求められ，そのテスト工程改善への要求は大きい．一方，モデルベーステストでは，モデルを作成し，そのモデルからテストケースを自動生成するテスト手法であり，テストの信頼性や作業効率の向上に寄与すると考えられている．そこで，本論文では，CAN 仕様のモデル化およびそのモデルからのテストケース生成法について論じ，その後，現在の進捗として，CAN 仕様のモデルを提示する．

1　はじめに

1.1　社会的背景

商品提供のサイクルの短期化と消費者の製品への増大により，本論文で扱う CAN(Controller Area Network) 製品をはじめとして，製品開発の現場では，開発期間短縮や製品の信頼性向上が課題となっている．この状況を改善するために，様々な活動が行われており，それらの中のひとつに，設計上流での品質の作りこみがある．

1.2　課題

設計上流での品質を作りこむ際の問題点として，設計上流での仕様の不具合 (曖昧さ，不整合，暗黙の仮定など) が設計へ与える影響が大きいこと，仕様の記述言語が自然言語なので個人で理解のレベルが異なり，仕様が正しく定義できないことなどが挙げられる．また，依然，開発期間の短縮という課題が残る．

1.3　目的

このような課題を解決するための手法のひとつに形式手法がある．設計上流での仕様を形式的仕様記述言語により記述することで，仕様の曖昧さや暗黙の仮定が低減し，さらに，形式的仕様 (モデル) を基にモデル検査や定理証明などの形式的検証を行うことで，仕様の不整合を検出できるようになる．実際，CAN 製品への形式手法適用に関して，形式的仕様記述言語による仕様記述 [3] や，モデル検査による検証 [7] [8] などの先行研究がある．しかし，モデル検査や定理証明などの形式的検証は，従来のテスト手法と大きく異なるので，現場への導入障壁が高いと考えられている．

以上の考察から，著者らは，形式的検証ではなく，モデルベーステスト (Model-Based Testing) [11] と呼ばれる形式手法を採用することにした．モデルベーステストは，テストの一種であり，モデルからテストケース生成がその特徴である．設計上流の仕様を形式化 (モデル化) することで，前出の課題が低減でき，また，そのモデルを用いてテストケースを自動生成するので，従来のテスト工程との親和性も高い．

この方針に基づき，著者らは，CPU 命令セット仕様のモデル化と，そのモデルからのテストケース自動

An Approach to Applying Model-Based Testing to CAN Products.

Keishi Okamoto, Yoshiki Kinoshita, 産業技術総合研究所 組込みシステム技術連携研究体, Collaborative Research Team for Verification, AIST.

Yuji Izumi, Satoshi Nakano, Toru Shimizu, ルネサスエレクトロニクス株式会社 技術開発本部, Technology Development Unit, Renesas Electronics Corporation.

生成に関する研究開発を行い，この手法が有効であると判断した [1]. 本論文では，研究開発 [1] を発展させ，CPU よりも MCU(Micro Control Unit) 製品開発での開発頻度が高い CAN 製品へのモデルベーステストの適用の取り組みを紹介する.

2 CAN プロトコル

2.1 CAN とは

CAN は車載ネットワーク用通信プロトコルとして開発されたが，現在では，産業機械やファクトリーオートメーションなどでも利用されるなど，その利用範囲は大きい.

CAN の仕様は，元々は Bosch 社による仕様 “CAN Specification Version 2.0” [4] であったが，現在では，ISO 11898 シリーズとして規格化されている．その中でも，本論文が対象としているデータリンク層の仕様は，“ISO 11898-1 Road vehicles - Cotroller area network (CAN) - Part 1: Data link layer and physical signalling” [6] として規格化されている．また，仕様以外にも，[9] [10] などの文献もある.

2.2 CAN の主な特徴

ISO の仕様をはじめ，多くの文献で CAN の特徴が挙げられているが，ここでは，ISO の仕様中の「6.1 CAN properties」で挙げられている CAN の特徴を以下に紹介する：

1. multi-master priority-based bus properties;
2. non-destructive contention-based arbitration;
3. multicast frame transfer by acceptance filtering;
4. remote data request;
5. configuration flexibility;
6. system-wide data consistency;
7. error detection and error signalling;
8. automatic retransmission of frames that have lose arbitration or have been destroyed by errors during transmission;
9. distinction between temporary errors and permanent failures of nodes and autonomous switching-off of defective nodes.

本論文では，CAN のデータリンク層を対象としており，データリンク層で見た場合，CAN の通信はフレームを論理的な単位として行われる．そこで，フレームについても紹介する．フレームの種類にはデータフレーム，リモートフレーム，エラーフレームなどがあり，例えば，データフレームは SOF(start of frame)，アービトレーション・フィールド (arbitration field) コントロール・フィールド (control field) などのフィールドにより構成され，さらに，例えば，アービトレーション・フィールドは ID(identifier) と RTR(remote transmission request) により構成される.

CAN の特徴 2 で挙げた arbitration(調停) とは，同時に送信要求が発生した場合に，どの送信要求中 CAN ノードに送信権があるかを決定することであり，送信要求中のノードが送信したフレームの ID，RTR の値を比較して行われる．また，7 で挙げた error signalling(エラー通知) は，エラーフレームを送信して行われる.

3 モデルベーステスト適用の枠組み

この章では，CAN 製品へのモデルベーステスト適用の枠組みを提案し，その枠組みの実現へ向けた取り組みを紹介する.

3.1 枠組みの提案

モデルベーステストの主要課題は，モデルの構築とそのモデルからのテストケース生成法の構築である．したがって，CAN 製品へのモデルベーステスト適用の枠組みを提案するためには，CAN の仕様のモデル化の枠組みとそのモデルからのテストケース生成法の枠組みを提案すればよい.

テストケース生成法を考えるためには，テストケースを定義する必要があり，モデルの構造を決めるためにも，そこから生成されるテストケースを定義する必要がある．CAN 製品のテストでは，CAN ノードへのいくつかフレームを入力し，それに対応して出力されるはずのフレームと実際に出力されたフレームを比較する．したがって，フレームを基本単位とする場合，入力するフレームの列とそれに対応して出力されるはずのフレームの列の組がテストケースとなる.

さらに，入力するフレームは動作“ 入力 ”と入力されるフレームの値の二つの情報に分割でき，同じく，出力されるはずのフレームは動作“ 出力 ”と出力されるはずのフレームの値の二つの情報に分割できるので，動作とフレームの値の組の列をテストケースと考えることが出来る．したがって，CAN ノードの動作と動作でやり取りされる値の列を生成可能なモデルが必要になる．

このような列を生成するためには，CAN ノードの動作に関する記述が必要である．そこで，モデル記述言語の候補として，モデル検査用モデル記述言語 Promela[5] とプロセス代数 ACP(the Algebra of Communicating Processes)[2] の二つを検討した．その結果，Promela による記述と従来の記述に類似性があることから，ACP を採用することにした．その理由は，大きく異なる二種類の記述に基づく二つの生成法を用いて，同じ目的のためのテストケースを生成し，両者の結果を比較することは，本論文で提案するモデルの妥当性検証に役立つと考えたからである．

ACP を用いて CAN ノードの動作を等式として記述 (モデル化) し，その等式を近似的に解くことで，その CAN ノードが取りうる基本動作 (basic action) の有限列の集まりが得られる．この基本動作の有限列を構成する個々の基本動作は，“ 上位層からフレーム f を受信する ”や“ 上位層へフレーム f' を送信する ”などを表す．したがって，等式を近似的に解いて得られる基本動作の有限列をテストケースと考えることが出来る．

以上の議論から，以下の枠組みを提案する：

- モデル：プロセス代数 ACP による CAN ノードの動作を表す等式
- テストケース生成：上記等式の近似解の生成

3.2 CAN 仕様のモデル化

3.2.1 モデル化のための仮定

本論文では，テスト技術者が CAN 製品のテスト項目を優先順位付けし，その結果に基づき，モデル化する対象を調停機能とエラー検出・通知機能に限定することにした．これにより，データフレーム，リモートフレームとエラーフレームのみをモデル化すればよい．また，モデルを単純にするため，ビットではなく，フレームを基本単位とすることにした．

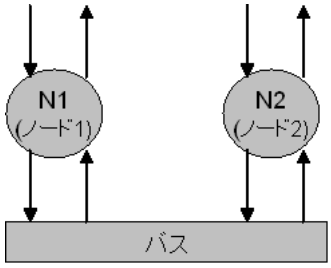

図 1 CAN ノードとバスの関係

さらに，モデルを単純にするため，バスに接続される CAN ノードの数は二とし (図 1)，ひとつの CAN ノードはバスとの通信用に読み込み用バッファと書き込み用バッファを持ち，上位との通信用に読み込み用バッファと書き込み用バッファを持つとする．さらに，各バッファは一フレームだけ格納できるとする．

3.2.2 基本動作の決定

プロセス代数 ACP によるモデル化を行うため，はじめに基本動作を表す記号を用意する必要がある．そこで，CAN ノードの動作を解析し，基本動作として必要なものを洗い出す．

CAN ノードの基本動作は，以下の三つのいずれかである：

1. 上位層からのフレーム受信,
2. 上位層へのフレーム送信,
3. バスとのフレームの送受信，エラー検出とフィルタリング.

実際は，“ バスとのフレームの送受信，エラー検出とフィルタリング ”はビット単位で行われているので，これらの動作は同時に起こると考えてよい．そこで，これらの動作をまとめて，ひとつの基本動作とした．なお，エラー通知 (エラーフレームの送信) も，これらの動作と不可分に実行されるが，フレーム単位を基本としてモデル化するために，エラー通知はこれらの一連の動作と分け，基本動作 3 のひとつとした．

同じ CAN ノードの受信でも，フレーム f の受信とフレーム f' の受信は，異なる基本動作として扱うべきである．そこで，フレームの動く範囲を定義する．本論文では，フレームは，データフレーム，リ

モートフレームとエラーフレームのみを考えている．エラー検出をテストしないのであれば，CAN 仕様書に従うもののみをフレームとして考えればよい．しかし，本論文では，エラー検出・通知をテストするために，壊れたフレームも許す必要がある．そこで，データフレームとリモートフレームの動く範囲全体 F を

$$F = \{\, f \mid f \text{ は長さ } 44 \text{ のビット列} \,\}$$

とする．(以下，特に断らない限り，$f, f', \ldots$ は F の要素であるとする) また，仮定により，エラーフレームはひとつだけであるので，それを e とする．さらに，便宜上，空のフレームを表す $\perp$ を用意する．例えば，基本動作 $n_1(\perp, f)$ は，CAN ノード $N1$ はバスへ何も送信しないが，バスからフレーム f を受信することを表す．なお，フレーム e と $\perp$ は壊れないと仮定する．

以上の考察から，以下の基本動作を用意する．なお, “：”の右側は左側の基本動作の直感的意味を表す：

定義 [基本動作] $(i = 1, 2)$

1. $r_i(f)$：CAN ノード N_i は上位層から f を受信,
2. $s_i(f)$：N_i は上位層へ f を送信,
3. $n_i(f, f')$：N_i はバスへ f を送信し，バスから f' を受信し，f, f' に基づきエラー検出し，f' をフィルタリングする,
4. $n_i(e, e)$：N_i はバスへエラーフレーム e を送信し，バスから e を受信する,
5. $n_i(\perp, e)$：N_i はバスへ何も送信せず，バスから e を受信する,
6. $c(f, f', f'', f''')$：$n_1(f, f')$ と $n_2(f'', f''')$ を合成したもの,
7. $c(e, e, \perp, e), c(\perp, e, e, e), c(e, e, e, e)$：6 と同様.

なお，エラーフレーム e は壊れないと仮定したので，バスへ e を送信したら，必ず e 自身を受信する．したがって，$n_i(e, f)$ の形の基本動作は考えなくてよい．

3.2.3 基本動作の合成

プロセス代数 ACP では，相互作用のある基本動作の組対し，合成した動作を定義する．そこで，上で用意した基本動作の組のうち，通信による相互作用がある組に対して，それらの合成を定義する．

定義 [基本動作の合成]

1. $c(f, f', f'', f''') = \gamma(n_1(f, f'), n_2(f'', f'''))$,
2. $c(\perp, f', f'', f''') = \gamma(n_1(\perp, f'), n_2(f'', f'''))$,
3. $c(f, f', \perp, f''') = \gamma(n_1(f, f'), n_2(\perp, f'''))$,
4. $c(e, e, \perp, e) = \gamma(n_1(e, e), n_2(\perp, e))$,
5. $c(\perp, e, e, e) = \gamma(n_1(\perp, e), n_2(e, e))$,
6. $c(e, e, e, e) = \gamma(n_1(e, e), n_2(e, e))$,
7. 上記以外の基本動作の合成は未定義 δ とする.

エラー検出・通知に関するテストを行わない場合，フレームが壊れるという仮定は無用となる．このとき，合成可能な基本動作の数は，上記の定義よりも減る．例えば，f が調停に勝つようなフレーム f と f' を考える場合，$n_1(f, f)$ と $n_2(f', f')$ は同時に起こりえない (調停により $N1$ が勝つので，受信されるフレームは必ず f である) ので，これらの動作は合成は未定義 δ となる．

3.2.4 CAN ノードの仕様

本小節では，CAN ノードの動作に関する仕様を，プロセス代数 ACP の等式として記述する．

CAN ノードの動作に関する仕様を，ACP を用いて記述する前に，CAN ノードの取りうる動作を木により表現すると分かりやすい．例えば，図 2 は CAN ノード $N1$ の動作を表している．このような図を $N1^{\perp}_{f}$，$N1^{f}_{\perp}$ や $N2$ などに対しても作成し，それらの図を基に等式を記述する．なお，例えば，図 2 中の $N1^{\perp}_{f}$ は，$N1$ の上位層への送信用バッファは空で，バスへの送信用バッファにはフレーム f が格納されているという $N1$ の状態を表している．$N1 \vdash f$ や $f' \in E$ などの定義は，下の定義を参照していただきたい．

ACP による記述のために，以下の表記を用意する：

定義 [等式記述のための表記]

- $f \leq f' \Leftrightarrow f$ と f' は調停の結果，f が勝つ,
- $f > f' \Leftrightarrow f$ と f' は調停の結果，f' が勝つ,
- $(f, f') \in E \Leftrightarrow (f, f')$ は (送信時のエラー検出対象である) ビット・エラー，スタッフ・エラー，CRC エラー，フォーム・エラー，アクノリッジ・エラーのいずれかを引き起こす,
- $f' \in E \Leftrightarrow f'$ は (受信時のエラー検出対象である) スタッフ・エラー，CRC エラー，フォーム・エラーのいずれかを引き起こす,.
- $Ni \vdash f \Leftrightarrow$CAN ノード Ni は f をフィルタリン

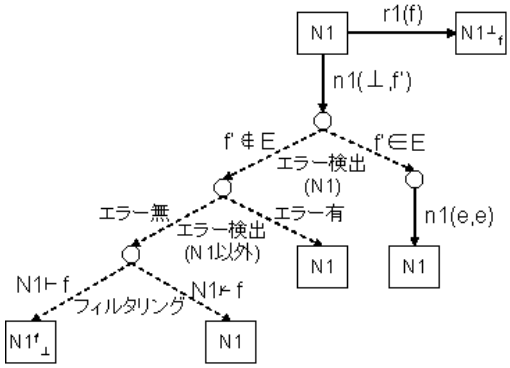

図 2 CAN ノード $N1$ の動作の木による表現

グして，受け入れる，

- $Ni \nvdash f \Leftrightarrow Ni$ は f をフィルタリングして，拒否する．

これらの表記を用いて，CAN ノード $N1$ の動作に関する仕様を ACP により記述する ($N2$ も同様な等式になる)：

$$
\begin{aligned}
N1 &= \Sigma_f r_1(f) * N1_f^{\perp} \\
&+ \Sigma_{f' \in E} n_1(\perp, f') * n_1(e,e) * N1 \\
&+ \Sigma_{f' \notin E} n_1(\perp, f') * n_1(\perp, e) * N1 \\
&+ \Sigma_{f' \notin E, N1 \nvdash f'} n_1(\perp, f') * N1 \\
&+ \Sigma_{f' \notin E, N1 \vdash f'} n_1(\perp, f') * N1_{\perp}^{f'}
\end{aligned}
$$

$$
\begin{aligned}
N1_{\perp}^{f'} &= \Sigma_f r_1(f) * N1_f^{f'} \\
&+ s_1(f') * (N1 + N1_{\perp}^{f'}) \\
&+ \Sigma_{f \in E} n_1(\perp, f) * n_1(e,e) * N1_{\perp}^{f'} \\
&+ \Sigma_{f \notin E} n_1(\perp, f) * n_1(\perp, e) * N1_{\perp}^{f'} \\
&+ \Sigma_{f \notin E, N1 \nvdash f} n_1(\perp, f) * N1_{\perp}^{f'} \\
&+ \Sigma_{f \notin E, N1 \vdash f} n_1(\perp, f) * N1_f^{f'}
\end{aligned}
$$

$$
\begin{aligned}
N1_f^{\perp} &= \Sigma_{f''} r_1(f'') * N1_f^{\perp} \\
&+ \Sigma_{f > f', f' \in E} n_1(f, f') * n_1(e,e) * N1_f^{\perp} \\
&+ \Sigma_{f > f', f' \notin E} n_1(f, f') * n_1(\perp, e) * N1_f^{\perp} \\
&+ \Sigma_{f > f', f' \notin E, N1 \nvdash f'} n_1(f, f') * N1_f^{\perp} \\
&+ \Sigma_{f > f', f' \notin E, N1 \vdash f'} n_1(f, f') * N1_f^{f'} \\
&+ \Sigma_{f \leq f', (f,f') \in E} n_1(f, f') * n_1(e,e) * N1_f^{\perp} \\
&+ \Sigma_{f \leq f', (f,f') \notin E} n_1(f, f') * n_1(\perp, e) * N1_f^{\perp} \\
&+ \Sigma_{f \leq f', (f,f') \notin E} n_1(f, f') * N1
\end{aligned}
$$

$$
\begin{aligned}
N1_f^{f'} &= \Sigma_{f''} r_1(f'') * N1_f^{f'} \\
&+ s_1(f') * (N1_f^{\perp} + N1_f^{f'}) \\
&+ \Sigma_{f > f'', f'' \in E} n_1(f, f'') * n_1(e,e) * N1_f^{f'} \\
&+ \Sigma_{f > f'', f'' \notin E} n_1(f, f'') * n_1(\perp, e) * N1_f^{f'} \\
&+ \Sigma_{f > f'', f'' \notin E, N1 \nvdash f''} n_1(f, f'') * N1_f^{f'} \\
&+ \Sigma_{f > f'', f'' \notin E, N1 \vdash f''} n_1(f, f'') * N1_f^{f'} \\
&+ \Sigma_{f \leq f''(f,f'') \in E} n_1(f, f'') * n_1(e,e) * N1_f^{f'} \\
&+ \Sigma_{f \leq f''(f,f'') \notin E} n_1(f, f'') * n_1(\perp, e) * N1_f^{f'} \\
&+ \Sigma_{f \leq f''(f,f'') \notin E} n_1(f, f'') * N1_{\perp}^{f'}
\end{aligned}
$$

一般に，作成されたモデルは，その妥当性を検証されるべきである．本論文におけるモデルの妥当性とは，このモデルから，CAN 製品のためのテストケースの十分な集まりを生成できることである．したがって，この場合の妥当性検証は，テストケース生成器が生成したテストケースの集まりを評価することであるが，それには，テストケース生成器作成後である研究開発後期まで待たなくてはならない．しかし，早期にモデルの妥当性を検証することは重要である．そこで，テスト技術者が，モデルが今回の目的に必要な CAN 仕様を満たしていることをレビューにより確認することをもって，モデルが妥当であるとすることとした．これにより，モデルは仕様に従った動作をすると期待でき，したがって，仕様と異なる動作により生成される不適切なテストケースの混入の可能性を低減できる．

3.3 テストケース生成の自動化

CAN ノード $N1$ と $N2$ の動作に関する等式を近似的に解けば，それぞれのノードが取りうる基本動作の列 (基本動作を “$*$” で結んだ基本項 (basic term)) の集まりが得られる．また，$N1$ と $N2$ を非同期合成したプロセス $N1||N2$ に，encapsulation 演算子 δ_H を作用させて得られるプロセス $\delta_H(N1||N2)$ に関する等式を近似的に解けば，$N1$ と $N2$ を非同期合成したシステムが取りうる基本動作の列，すなわちテストケース，の集まりが得られる．なお，H は以下で定義する：

$$H = \{\, n_1(f, f'), n_2(f, f') \mid f, f' \in F \cup \{\, e, \perp \,\} \,\}$$

$N1$ がフレーム f をバスへ送信して，調停に勝利した場合，f が壊れなければ，$N2$ も f をバスから同時に受信する．したがって，$\delta_H(N1||N2)$ の動作に関する等式を近似的に解いて得られる基本動作の列の集まりの中に，基本動作 $n_1(f,f)$ が単独で現れることはなく，$(n_1(f,f)|n_2(f',f))$ の形で現れるはずである．実際，H の定義から，$\delta_H(n_1(f,f)) = \delta$ となるので，$r_1(f) * r_2(f') * n_1(f,f) * n_2(f',f) * s_2(f)$ のような列は $\delta_H(N1||N2)$ の近似解の集まりの中に現れない．

$\delta_H(N1||N2)$ の動作に関する等式の近似解として得られる基本動作の列がテストケースであることを理解していただくために，基本動作の列の例を挙げる．

例 1：$f \leq f', N2 \vdash f$ とする．

$$r_1(f) * r_2(f') * (n_1(f,f)|n_2(f',f)) * s_2(f)$$

例 1 では，$N1$ が上位層から f を受信し，$N2$ が上位層から f' を受信する．その後，それぞれがバスへ f と f' を送信すると同時に，調停に勝った f を受信する．最後に，$N2$ が受け入れた f を上位層へ送信する．この列は，調停機能をテストするためのテストケースのひとつである．

例 2：$f \leq f', (f,f') \in E$ とする．

$$r_1(f) * (n_1(f,f')|n_2(\bot,f')) * (n_1(e,e)|n_2(\bot,e))$$

例 2 では，$N1$ が上位層から f を受信する．その後，$N1$ が f をバスへ送信し，$N2$ は何もバスへ送信しないと同時に，何らかの理由で f が壊れて f' となり，両ノードは f' をバスから受信する．このとき，$N1$ は，f' の調停フィールドの内容から，調停に勝ったと認識しているが，f, f' の値から，$N1$ は送信時のエラーを検出する．さらにその後，$N1$ がエラーフレーム e を送信し，$N2$ が e を受信するという動作が同時に起こる．この列は，エラー検知・検出機能をテストするためのテストケースのひとつである．

以上の例示から，$\delta_H(N1||N2)$ の動作に関する等式の近似解として得られる基本項の中にテストケースとして適切な基本動作の列があることが分かった．

4 まとめ

本論文の目的は，CAN 製品へのモデルベーステストの適用の取り組みを紹介することであった．そこで，本論文での取り組みをここにまとめ，これまでの取り組みで分かった問題点について述べる．

4.1 モデルベーステスト適用の取り組みのまとめ

取り組みは，以下の作業に細分化される．1) はじめに，CAN 製品へのモデルベーステストの適用に必要な知識を得るために，CAN の仕様や関連文献を調査した．2) 次に，テスト技術者の専門知識を基に，テストケースを CAN ノードの動作の列として定義した．3) その後，モデル記述言語としてプロセス代数 ACP を選択し，決められたデータ構造を持つテストケースを生成することができるモデル (CAN ノード $N1$ と $N2$ の動作に関する等式) を構築した．4) 等式の近似解 (基本動作の有限列) がテストケースであるが，モデルを用いて，プロセス $\delta_H(N1||N2)$ を有限回展開すれば，テストケースが得られる．残る課題は，構築したモデルを入力とするテストケース生成器の作成である．

4.2 モデル化方針の問題点

CAN の仕様中の「10.9 Error signalling」で記述されているように，多くの場合，CAN ノードはフレームの受信中のエラーを検出した直後に，エラーフレームの送信を開始する．他方，本論文のように一フレームの送受信を基本動作とする場合，受信中にエラーを検出したフレームの受信完了後に，エラーフレームを送信することになり，実際の動作と異なってしまう．

実際，プロセス $\delta_H(N1||N2)$ の動作に関する等式の近似解を求めると，$f \leq f', f \notin E, f' \in E, N2 \vdash f$ であるような f と f' に対し，以下の二つの基本動作の列が得られる．

1. $r_1(f) * (n_1(f,f')|n_2(\bot,f)) * (n_1(e,e)|n_2(\bot,e))$
2. $r_1(f) * (n_1(f,f')|n_2(\bot,f)) * s_2(f)$

CAN の仕様に忠実な動作では，$(n_1(f,f')|n_2(\bot,f))$ を処理中に，$N1$ がエラーを検出し，$(n_1(e,e)|n_2(\bot,e))$ へと処理が移る (1 の動作の列)．しかし，本論文のモ

デル化では，エラーフレームの送受信を別の基本動作としているために，$N1$ によるエラー検出の直後に，別の基本動作 $s_2(f)$ が実行される (2 の動作の列) という，本来はありえない動作も許してしまう．

この問題は，一フレームの送受信を基本動作としていることに起因しているが，実際の仕様と同様に，一ビットの送受信を基本動作とすることで解決できる．しかし，一ビットの送受信を基本動作とした場合，生成される近似解 (テストケース) の数が膨大になるという別の問題が発生する．

5 将来課題

5.1 生成されたテストケースの集まりの意味

3 章でも述べたように，テストケースはプロセス $\delta_H(N1||N2)$ の動作に関する等式の近似解として生成される．しかし，単純に等式の近似解として生成したテストケースの集まりが，どんな性質をテストするためのものかは不明である．そこで，近似解生成の際に，テストしたい項目を入力すると，そのテスト項目のためのテストケースの集まりのみを生成するように，近似解生成を制限する工夫が課題となる．

5.2 テスト技術者の専門知識に基づく展開制限

等式の近似解として得られるテストケースを全て生成することは現実的でない．近似解の計算の段階で，適切な制約を組み込むことで，生成されるテストケースを制限すべきである．この方法ならば，例えば，不適切な動作の列 s をイニシャルセグメントとする列 s' は，s が不適切と判断された段階で自動的に排除できる．

現実のテストでは，テスト技術者の専門知識に基づき，生成するテストケースを制限している．そこで，本研究開発でも同様に，テスト技術者の専門知識に基づき，等式の近似解の計算を制限するという方針が考えられる．この方針であれば，生成されるテストケースの集まりは，従来のテストケースの集まりと同じものになるはずである．したがって，潜在化している不具合数の推定などで，過去のテスト資産を活かすことが出来ると考える．

謝辞 ルネサスエレクトロニクスの上村俊之様には，CAN に不慣れな産総研研究者に対し，CAN の丁寧な解説をしていただいたことを感謝いたします．なお，本研究開発は，科学技術振興機構の平成 22 年度研究成果最適展開支援事業の一環として行われています．

参 考 文 献

[1] Abe, T., Higuchi T., Imai R., Kinoshita Y., Nakano S., Okamoto K., Saito M. and Takeyama M.：Formalization of System LSI Specification and Automatic Generation of Verification Items -Extended Abstract-, *TESTCOM/FATES 2008 Supplementary Proceedings, Edited by Kenji Suzuki, Teruo Higashino, Andreas Ulrich, Toru Hasegawa*, 2008.

[2] Baeten, J.C.M. and Weijland, W.P.：*Process Algebra*, Cambridge Tracts in Theoretical Computer Science 18, Cambridge University Press, 1990.

[3] Benzekri, A. and Bruel, J.M.：Controller area network: a formal case study, *Factory Communication Systems, 1997. Proceedings*, IEEE International Work, 1997.

[4] Robert Bosch GmbH：BOSCH CAN Specification 2.0, 1991.

[5] Holzmann, G.J.:THE SPIN MODEL CHECKER, Addison-Wesley, 2004.

[6] ISO 11898-1 Road vehicles - Controller area network (CAN) - Part 1: Data link layer and physical signalling, 2003.

[7] Leen, G. and Hefferman, D.：Modeling and Verification of a Time-triggered Networking Protocol, *Proceedings of ICNICONSMCL'06*, IEEE Computer Society, 2006.

[8] van Osch, M.P.W.J.:Finite State Analysis of the CAN bus Protocol "What CAN Can and Can Not do", MASTER'S THESIS, 2001.

[9] ルネサスエレクトロニクス：CAN 入門書, 2006.

[10] 佐藤道夫：車載ネットワーク・システム徹底解説, CQ 出版, 2005.

[11] Utting, M. and Legeard, B.：Practical Model-Based Testing: A Tools Approach, Morgan Kaufmann, 2006.