数理解析研究所講究録
第1849巻 2013年 64-70

動的ハイブリッドCEGAR検証器の開発

柳瀬 龍* 酒井 辰典† 酒井 誠‡ 山根 智§

1 はじめに

組込みシステムに対する安全性検証の有効な手法として，ハイブリッドオートマトンに基づく仕様記述およびモデル検査が挙げられる [1, 2]. しかし，モデル検査における状態空間の探索は網羅的に行うために，システムの規模が大きくなったとき状態爆発問題が大きな課題となる [3]. この問題を回避する手段の1つとして，E. M. Clarke らの提案した反例による抽象化精錬 (Counterexample Guided Abstraction Refinement; CEGAR) の枠組みの適用が挙げられる [4, 5, 6]. 本稿では，仕様記述言語として動的線形ハイブリッドオートマトン (DLHA) を定義する. また，これに CEGAR の枠組みを適用した動的ハイブリッド CEGAR を考え，安全性検証のためのモデル検査手法を述べる.

2 動的線形ハイブリッドオートマトン (DLHA)

動的線形ハイブリッドオートマトン (Dynamic Linear Hybrid Automaton; DLHA) は，オートマトンの生成・消滅およびキュー操作のアクションにより拡張された線形ハイブリッドオートマトンである. ここでは，DLHA の構文と意味を定義する.

2.1 DLHA の構文

DLHA H は，8 つの要素からなる組 $(L, Var, Inv, Flow, Act, T, t_{init}, T_{end})$ で表される. ここで，

- ロケーションの有限集合 L.
- 実数変数の有限集合 Var. また，各変数 $x \in Var$ に実数を割り当てる関数 ν を変数の評価といい，$\nu(x) \in \mathbb{R}$ を変数評価値という. 変数評価の集合を V と表す.
- 各ロケーション $l \in L$ に不変条件 ϕ を割り当てる関数 Inv. ϕ は変数の制約条件であり，以下のように定義される.

$$\phi \stackrel{\text{def}}{=} true \mid asap \mid x \sim e \mid x - x' \sim e \mid \phi_1 \wedge \phi_2$$

ただし，$x, x' \in Var, e \in \mathbb{Q}$, ϕ_1, ϕ_2 は制約条件，$\sim \in \{<, \leq, >, \geq, ==\}$ である. また，すべての制約条件の集合を $\Phi(Var)$ と書く.

- 各ロケーション $l \in L$ にフロー条件 f を割り当てる関数 $Flow$. フロー条件 f は以下のように定義される.

$$f \stackrel{\text{def}}{=} \dot{x} = c \mid f_1 \wedge f_2$$

ただし，$x \in Var, c \in \mathbb{Q}$, f_1, f_2 はフロー条件. また，すべてのフロー条件の集合を $F(Var)$ と書く.

- アクションの有限集合 $Act = Act_{in} \cup Act_{out} \cup Act_{\tau}$, ここで，$Act_{in}$ は入力アクションの有限集合，Act_{out} は出力アクションの有限集合，Act_{τ} は内部アクションの有限集合である. 特に，$Crt\_H_n! \in Act_{out}, Crt\_H_n? \in Act_{in}$ はオートマトン H_n を生成するアクション，$Dst\_H_n! \in Act_{out}, Dst\_H_n? \in Act_{in}$ はオートマトン H_n を消滅させるアクション，$Q!H_n \in Act_{\tau}$ はオートマトン H_n の生成要求をキューに格納するアクション，$Q?H_n \in Act_{\tau}$ はオートマトン H_n の生成要求をキューから取り出すアクションとする.

*金沢大学大学院自然科学研究科
†第1著者に同じ
‡第1著者に同じ
§金沢大学理工研究域

- 遷移関係の有限集合 $T \subseteq L \times \Phi(Var) \times Act \times 2^{UPD(Var)} \times L$. このとき,
  - $\phi \in \Phi(Var)$ はガード条件.
  - $UPD(Var)$ は算術式の有限集合.
  - 算術式 $upd \in UPD(Var)$ は以下のように定義される.

    $$upd \stackrel{\text{def}}{=} x := const \mid x := x + const$$

    ただし, $x \in Var$, $const \in \mathbb{Z}$.
- 初期遷移 $t_{init} \in L \times (Act_{in} \cup Act_{\tau}) \times 2^{UPD(Var)}$.
- 破棄遷移の集合 $T_{end} \subseteq L \times \Phi(Var) \times (Act_{out} \cup Act_{\tau})$.

2.1.1 DLHA の例

DLHA の例を図 1 に示す. ここで, DLHA $\mathcal{A}_A$ および $\mathcal{A}_B$ は,

$$\mathcal{A}_A = (L_A, Var_A, I_A, F_A, Act_A, T_A, t_{init_A}, T_{end_A}),$$
$$\mathcal{A}_B = (L_B, Var_B, I_B, F_B, Act_B, T_B, t_{init_B}, T_{end_B}).$$

ただし,

$$L_A = \{\text{Off}, \text{On}\},\ Var_A = \{c_A\},$$
$$I_A : \{\text{Off} \mapsto c_A \leq 10,\ \text{On} \mapsto c_A \geq 0\},$$
$$F_A : \{\text{Off} \mapsto \dot{c_A} = 1,\ \text{On} \mapsto \dot{c_A} = 0\},$$
$$Act_A = \{Crt\_\mathcal{A}_B!, Dst\_\mathcal{A}_B?, start\},$$
$$T_A = \{(\text{Off}, c_A == 10, Crt\_\mathcal{A}_B!, \{\}, \text{On}),$$
$$(\text{On}, true, Dst\_\mathcal{A}_B?, \{c_A := 0\}, \text{Off})\},$$
$$t_{init_A} = (\text{Off}, start, c_A := 0),\ T_{end_A} = \{\},$$

であり,

$$L_B = \{\text{Exec}\},\ Var_B = \{c_B\},$$
$$I_B : \{\text{Exec} \mapsto c_B \leq 5\},$$
$$F_B : \{\text{Exec} \mapsto \dot{c_B} = 1\},$$
$$Act_B = \{Crt\_\mathcal{A}_B?, Dst\_\mathcal{A}_B!\}, T_B = \{\},$$
$$t_{init_B} = (\text{Exec}, Crt\_\mathcal{A}_B?, c_B := 0),$$
$$T_{end_B} = \{(\text{Exec}, c_B == 5, Dst\_\mathcal{A}_B!)\}.$$

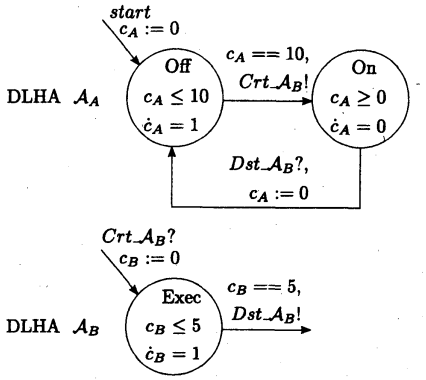

図 1: 動的線形ハイブリッドオートマトンの例

2.2 意味

2.2.1 キュー

キュー (無限長 FIFO バッファ) を $\mathcal{Q}$, キューに格納されうるメッセージの集合を M とする. このとき, キューの内容, すなわちキュー内のメッセージ列は

$$w_{\mathcal{Q}} \in M^*$$

で表される.

キューの操作に関するアクションは, キューを $\mathcal{Q}$ として $\mathcal{Q}!w$ もしくは $\mathcal{Q}?w$ の形で表される. ここで, $w \in M^*$ である.

2.2.2 DLHA の状態

DLHA の状態 σ は

$$\sigma \stackrel{\text{def}}{=} (l, \nu, w_{\mathcal{Q}})$$

と定義される. ここで,

- $l \in L$ はロケーション.
- $\nu \in \mathbb{R}_{\geq 0}$ は時間 t における変数評価値.
- $w_{\mathcal{Q}} \in M^*$ はキュー $\mathcal{Q}$ に格納されているメッセージ列.

2.2.3 DLHA の意味

DLHA $H = (L, Var, Inv, Flow, Act, T, t_{init}, T_{end})$ の意味 $\mathcal{M}$ は,

$$\mathcal{M} \stackrel{\text{def}}{=} (\Sigma, \Rightarrow, \sigma_0)$$

として定義される.

- Σ は状態 σ の集合.
- $\Rightarrow$ は時間遷移 $\Rightarrow_\delta$ と離散遷移 $\Rightarrow_d$ の和集合.
  - 時間遷移: 任意の状態 $(l, \nu, w_Q) \in \Sigma$ と時間経過 $t \in \mathbb{R}_{\geq 0}$ に対し, $l' = l$ かつ $\nu' = \nu + Flow(l) \cdot t \in Inv(l)$ の時かつその時に限り $(l, \nu, w_Q) \Rightarrow_\delta (l, \nu', w_Q)$ である.
  - 離散遷移: ある遷移関係において条件が満たされている場合, ロケーションに留まる不変条件を満たしていても即座に離散遷移するような動作として, ガード条件 $asap$ が定義されている. これは時間遷移のない瞬時的な離散遷移の動作を行う場合に有用となる.
    * アクション a が生成アクション及び消滅アクションでない時, 任意の状態 $(l, \nu, w_Q) \in \Sigma$ に対し, ロケーション l からのエッジ $(l, \phi_g, a, \lambda, l') \in T$ が存在し, ν が ϕ_g を満たし, かつ λ で更新された ν' が $\nu' \in Inv(l')$ となるとき $(l, \nu, w_Q) \Rightarrow_d (l', \nu', w_Q)$ である.
    * アクション a が生成アクションの時, 任意の状態 $(l_1, \nu_1, w_Q) \in Q$ に対し, ロケーション l_1 からのエッジ $(l_1, \phi, a, \lambda, (l_1', l_2)) \in T$ が存在し, ν_1 が ϕ_g を満たし, かつ λ で更新された ν_1' が $\nu_1' \in Inv(l_1')$ となるとき $(l_1, \nu_1, w_Q) \Rightarrow_d ((l_1', l_2), (\nu_1', \nu_2), w_Q)$ である.
    * アクション a が消滅アクションの時, 任意の状態 $((l_1, l_2), (\nu_1, \nu_2), w_Q) \in \Sigma$ に対し, ロケーション (l_1, l_2) からのエッジ $((l_1, l_2), \phi_g, a, \lambda, l_1') \in T$ が存在し, ν_1, ν_2 が ϕ_g を満たし, かつ λ で更新された ν' が $\nu' \in Inv(l')$ となるとき $((l_1, l_2), (\nu_1, \nu_2), w_Q) \Rightarrow_d (l_1', \nu_1', w_Q)$ である.
    * アクション a がメッセージをキューに格納するアクションの時, 任意の状態 $(l, \nu, w_Q) \in Q$ に対し, ロケーション l からのエッジ $(l, \phi_g, a, \lambda, l') \in T$ が存在し, ν が ϕ を満たし, かつ λ で更新された ν' が $\nu' \in Inv(l')$ となるとき $(l, \nu, w_Q) \Rightarrow_d (l', \nu', w_Q')$ である. ただし, $w_Q' = w_Q \cdot w$ である.
    * アクション a がメッセージをキューから取り出すアクションの時, 任意の状態 $(l, \nu, w_Q) \in Q$ に対し, ロケーション l からのエッジ $(l, \phi_g, a, \lambda, l') \in T$ が存在し, ν が ϕ_g を満たし, かつ λ で更新された ν' が $\nu' \in Inv(l')$ となるとき $(l, \nu, w_Q) \Rightarrow_d (l', \nu', w_Q')$ である. ただし, $w_Q' = w_Q \cdot w$ である. ただし, $w_Q = w \cdot w_Q'$ である.
- $\sigma_0 \in \Sigma$ は初期状態.

3 動的ハイブリッド CEGAR

3.1 検証器の構成

動的ハイブリッド CEGAR は大きく分けて, 抽象化・到達可能性解析, 反例解析, 精錬という工程からなる. 処理の概要を以下に示す.

- 抽象化・到達可能性解析
  - 抽象化: 元のモデル (以下, 具体モデル) に対して抽象化を行い, 抽象モデルを構築する. 本研究では, 抽象化の手法として述語抽象化を用いる.
  - 到達可能性解析: 構築された抽象モデル上で, 与えられた目的ロケーションに到達するかどうかを調べる.
- 反例解析: 到達可能性解析において目的ロケーションに到達するパス (抽象反例と呼ぶ) が存在

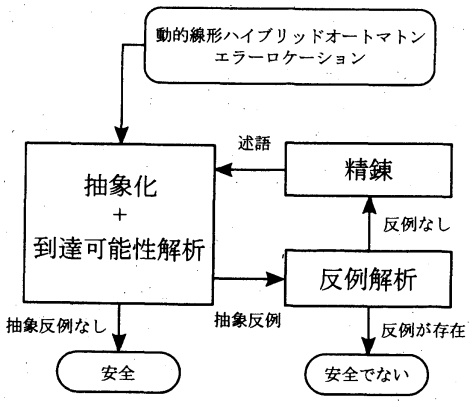

図 2: 動的ハイブリッド CEGAR 検証器の構成

したとき，そのパスが具体モデルにおいても存在するかどうかを調べる．抽象反例が具体モデルにおいても存在する場合には，到達可能として終了する．具体モデル上で存在しない場合には，精錬へ進む．このとき，具体モデル上に存在しない抽象反例を偽反例という．

- 精錬: 偽反例の元となるロケーションに加える抽象化述語を見つけ，再び抽象化へ進む．

3.2 抽象化・到達可能性解析

ここでは，抽象化及び到達可能性解析の手法を述べる．動的ハイブリッド CEGAR では，並列合成における状態爆発抑制のため抽象モデルを動的に構築しながら到達可能性解析を行う．

3.2.1 述語抽象化

述語抽象化は，状態爆発を抑制するために用いられる主な手法の 1 つである．述語を用いることで実数変数の変数評価に対する抽象化を行う．

抽象化述語 抽象化のための述語 ψ を

$$\psi \stackrel{\text{def}}{=} true \mid a_1x_1 + \cdots + a_nx_n + a_{n+1} \sim 0$$

と定義する．ただし，$x_1, \ldots, x_n \in Var$, $a_i \in \mathbb{Q}$ $(i \in \{1, \ldots n+1\})$, $\sim \in \{<, \leq, >, \geq\}$ である．

ロケーション l における抽象化述語の有限集合を $\Psi^l = \{\psi_0^l, \ldots, \psi_{n-1}^l\}$ とする．また，全てのロケーションにおける抽象化述語の有限集合を $\Psi = \{\Psi^{l_0} \cup \cdots \cup \Psi^{l_{n-1}}\}$ とする．

ロケーション l における抽象化述語の集合 $\Psi^l = \{\psi_0^l, \ldots, \psi_{n-1}^l\}$ は，l においてのみ成立する述語であり，サイクル評価 ν から長さ n のビットベクトル b^l への写像である．サイクル評価 ν に対する抽象化述語 ψ_i^l を $\psi_i^l\nu$ で表す．ここで，全てのロケーションにおける抽象化述語 Ψ により，抽象化関数 α が決まる．すべてのロケーションにおけるビットベクトルの集合を $\mathcal{B}$ とする．α の逆像は具体化関数 γ であり，ビットベクトルの集合からビットベクトルの i 番目の要素が真であるときは常に ψ_i^l を満たすような全てのサイクル評価の集合への写像である． 従って，具体状態 $(l, \nu, w_\mathcal{Q})$ の集合は抽象化関数 α により抽象状態 $\alpha((l, \nu, w_\mathcal{Q}))$ に変換され，抽象状態 $(l, b, w_\mathcal{Q})$ は γ により具体状態の集合 $\gamma((l, b, w_\mathcal{Q}))$ に写像される．

述語抽象化と具体化 Var を変数の有限集合として，$\mathcal{U}$ は対応する変数評価の集合とする．抽象化述語の有限集合 $\Psi^l = \{\psi_0^l, \ldots, \psi_{n-1}^l\}$ が与えられたとき，タイミング制約抽象化関数 $\alpha : L \times \mathcal{U} \times M^* \to L \times \mathcal{B} \times M^*$ は以下のように定義される．

$$\alpha((l, \nu, w_\mathcal{Q}))(i) = (l, \psi_i^l\nu, w_\mathcal{Q})$$

また，具体化関数 $\gamma : L \times \mathcal{B} \times M^* \to L \times 2^\mathcal{U} \times M^*$ は以下のように定義される．

$$\gamma((l, b, w_\mathcal{Q})) = \{(l, \nu, w_\mathcal{Q}) \in L \times \mathcal{U} \times M^* \mid I(l) \wedge \bigwedge_{i=0}^{n-1} \psi_i^l\nu \equiv b^l(i)\}$$

抽象化述語と抽象化関数，具体化関数を用いて抽象構造を構築する．抽象構造は，オーバー近似 (over-approximation) になるように構築する．オーバー近似とは，具体構造が持つ遷移を抽象構造に全て持たせることで，抽象化に健全性を持たせる近似である．

3.2.2 到達可能性解析

到達可能性解析では，抽象化を行いながら幅優先探索により状態空間の探索を行う．解析では，抽象反例が見つかった時点で反例解析を行う．手順は以下のようになる．

1. 各オートマトンの初期ロケーションを合成，抽象化し抽象初期状態 $\sigma_0^{\mathcal{A}}$ を求める．探索する状態を格納する待ち行列 *Queue* の末尾に $\sigma_0^{\mathcal{A}}$ を追加し，訪問した状態の集合 *Visit* および探索木 *Tree* に対して，$Visit \leftarrow \emptyset$, $Tree \leftarrow \emptyset$ とする．
2. *Queue* が空になるまで以下の処理を繰り返す．
   (a) *Queue* の先頭から抽象状態 $\sigma^{\mathcal{A}}$ を取り出し，$Visit \leftarrow Visit \cup \{\sigma^{\mathcal{A}}\}$，次状態の集合 $\Sigma_{post}^{\mathcal{A}} \leftarrow \emptyset$ とする．
   (b) $\sigma^{\mathcal{A}}$ が目的ロケーションを含む場合，抽象初期状態 $\sigma_0^{\mathcal{A}}$ から $\sigma^{\mathcal{A}}$ までのパスを探索木 *Tree* から求めて出力し，終了する．そうでなければ，次へ進む．
   (c) $\sigma^{\mathcal{A}}$ に含まれるロケーションから出る各遷移について以下の処理を行う．
      i. 遷移によって得られるすべてのロケーションを求め，合成する．
      ii. 合成したロケーションをもつ抽象状態が抽象モデルに含まれていなければ抽象状態 $\sigma_{post}^{\mathcal{A}}$ を新たに生成し，$\Sigma_{post}^{\mathcal{A}} \leftarrow \Sigma_{post}^{\mathcal{A}} \cup \{\sigma_{post}^{\mathcal{A}}\}$ とする．すでに抽象モデルに含まれていた場合には抽象化述語 Ψ に従って抽象モデルの再構築を行い，抽象化された状態の集合 $\Sigma_r^{\mathcal{A}}$ に対して $\Sigma_{post}^{\mathcal{A}} \leftarrow \Sigma_{post}^{\mathcal{A}} \cup \Sigma_r^{\mathcal{A}}$ とする．
   (d) $\Sigma_{post}^{\mathcal{A}} \leftarrow \Sigma_{post}^{\mathcal{A}} \setminus Visit$ として，$\sigma^{\mathcal{A}}$ から各抽象状態 $\sigma_{post}^{\mathcal{A}} \in \Sigma_{post}^{\mathcal{A}}$ に対し $Tree \leftarrow Tree \cup \{(\sigma^{\mathcal{A}}, \sigma_{post}^{\mathcal{A}})\}$，*Queue* の末尾へ $\Sigma_{post}^{\mathcal{A}}$ 追加する．
3. "*not reachable*" と出力し終了する．

3.3 反例解析

3.3.1 凸多面体

凸多面体は変数間の制約の連言によって記述された変数領域の集合であり，変数の集合 C と制約条件 Φ に対して決まる．形式的には，

$$\zeta = \bigwedge_{0 \le k \le e} (\ v_{1,k}\, x_1 + \ldots + v_{i,k}\, x_i \sim_k \ d_k)$$

と書ける．ここで，x_i は変数の集合 *Var* の要素，$0 \le i \le |Var|$．ただし，$x_0 = 0$ とする．$v_{j,k} \in \mathbb{Q}, e \in \mathbb{N}$ は式の総数，$d_k \in \mathbb{Q} \cup \{\infty\}$，$\sim_k \in \{<, \le\}$ である．

本稿では，凸多面体上の演算として以下の操作を用いる．

凸多面体上の演算

- 凸多面体の時間後退演算

$$\mathrm{Tpre}\ [l, \zeta] = \{\nu - Flow(l) \cdot t \mid \nu \in \zeta, t \in \mathbb{R}_{\ge 0}\}$$

- 凸多面体のリセット

$$\mathrm{Reset}\ [X, \zeta] = \{\nu[Var := 0]\}$$

ここで，X は変数の集合 $X \subseteq Var$ である．

- 凸多面体の制約除去

$$\mathrm{Free}\ [X, \zeta] = \mathcal{U}$$

ここで，X は変数の集合 $X \subseteq Var$，$\mathcal{U}$ は *Var* に対する変数評価の集合．

- 変数の追加

$$\mathrm{Add}\ [Var_{Add}, \zeta] = \bigwedge_{0 \le k \le i} (\ v_{1,k}\, x_1 + \cdots + v_{i,k}\, x_i \sim_k \ d_k)$$

例として，$Var = \{x\}, Var_{Add} = \{a\}$ のとき，$Var \cup Var_{Add} = \{x, a\}$ であり，$x_1 = x, x_2 = a$ である．追加される変数は各変数集合の後ろに追加され，領域は $a \le 0 \wedge -a \le 0$ というように領域が追加される．それに伴い $i = i + |Var_{Add}|$ となる．

- 変数の削除

$$\mathrm{Del}\,[Var_{Del},\zeta] = \bigwedge_{0\le k\le i}(\,v_{1,k}\,x_1+\cdots+v_{i,k}\,x_i \sim_k\ d_k)$$

例として，具体的には，$Var=\{x,c\}, Var_{Del}=\{c\},\zeta = x\le 5\wedge -x\le 0\wedge c\le 200\wedge -c\le 0\wedge 100x+c\le 200$ の時，$\{Var\}\setminus Var_{Del}=\{x\},\mathrm{Del}\,[Var_{Del},\zeta]=x\le 2\wedge -x\le 0$ となる．

抽象反例が具体モデル上で再現できるかを調べる．具体的には，抽象反例 $\omega_{ce}^{\mathcal{A}}$ を具体モデル上で辿り，各抽象状態の具体化がある，すなわち，$\Rightarrow\ \in \gamma(\Rightarrow^{\mathcal{A}})$ となるような元の遷移系における反例 $\omega_{ce}=\sigma_0\Rightarrow\cdots\Rightarrow\sigma_i\Rightarrow\cdots\Rightarrow\sigma_{target}$ が存在するかを判定する（ただし，$\sigma_i=(l_i,\zeta_i,w_{\mathcal{Q}})$）．計算法としては，ターゲットから後方に向かって，現在の状態に遷移可能な状態集合を求めていく．これは最弱前条件 [7] の考え方に基づいている．また，遷移に生成アクションがある場合，そのオートマトンは生成される前の構成になるため，使用されている変数を削除する．この時点の領域が削除対象となる変数がすべて 0 という領域を含まれなければ，抽象反例は再現できない偽反例である．

遷移に消滅アクションがある場合，そのオートマトンで使用される変数 Var に対して領域を $\mathcal{U}$ として追加する．これらの操作を行い，初期状態を含む集合が得られたら，反例であることが分かる．途中で空集合となったときは，その抽象反例は具体モデルでは再現できないので，偽反例であることが分かる．この概念を詳細に記述すると以下のようになる．

1. 最後尾のロケーションに到達することができる凸多面体が存在するかどうか求めるため，抽象反例の最後尾のロケーション l_{last} を述語 $true$ により述語抽象化されている状態 $(l_{last},true,w_{\mathcal{Q}_{last}})$ とみなし，具体化関数 γ を適用する．これによって，l_{last} の不変条件と述語により定められる凸多面体 ζ_{last} を用いた組 $(l_{last},\zeta_{last},w_{\mathcal{Q}_{last}})$ を得る．

2. $(l_{last},\zeta_{last},w_{\mathcal{Q}_{last}})$ から後方に，到達可能な最大の状態集合を求めていく．具体的には，$\sigma_i^{\mathcal{A}}$ を具体化関数 γ によって元の動的線形ハイブリッドオートマトン上の状態に戻し，その状態から出発する時点の凸多面体 ζ_i^{out} と，その状態に到達した時点の凸多面体 ζ_i^{in} を以下のように計算する．

   - 離散遷移で，生成・消滅アクションがない，またはキューアクションの遷移のとき，遷移のリセット λ_i，ガード条件 ϕ，遷移元ロケーションの述語 $b^{l_i}\Psi^{l_i}$，不変条件 $Inv(l_i)$ を用いて遷移先の凸多面体 ζ_{i+1}^{in} に到達可能な凸多面体 ζ_i^{out} を求める．

$$\zeta_i^{out} = Inv(l_i)\wedge b^{l_i}\Psi^{l_i}\wedge\phi_i \wedge \mathrm{Free}[\lambda_i,\zeta_{i+1}^{in}\wedge\mathrm{Reset}[\lambda_i,true]]$$

   - 離散遷移で，生成アクションがある遷移のとき，遷移のリセット λ_i，削除する変数 C_{Del}，遷移元ロケーションの述語 $b^{l_i}\Psi^{l_i}$，不変条件 $Inv(l_i)$ を用いて遷移先の凸多面体 ζ_{i+1}^{in} に到達可能な凸多面体 ζ_i^{out} を求める．

$$\zeta_i^{out} = Inv(l_i)\wedge b^{l_i}\Psi^{l_i}\wedge \mathrm{Del}[Var_{Del},\zeta_{i+1}^{in}\wedge\mathrm{Reset}[Var_{Del},true]]]$$

     ただし，$Var_{Del}\subset Var$ は生成されたオートマトンで使われている変数の集合である．

   - 離散遷移で，消滅アクションがある遷移のとき，遷移のガード条件 ϕ_i，追加する変数 Var_{Add}，遷移元ロケーションの述語 $b^{l_i}\Psi^{l_i}$，不変条件 $Inv(l_i)$ を用いて遷移先の凸多面体 ζ_{i+1}^{in} に到達可能な凸多面体 ζ_i^{out} を求める．

$$\zeta_i^{out} = Inv(l_i)\wedge b^{l_i}\Psi^{l_i}\wedge\phi_i \wedge \mathrm{Free}[Var_{Add},\mathrm{Add}[Var_{Add},\zeta_{i+1}^{in}]]$$

     ただし，$Var_{Add}\subset Var$ は消滅したオートマトンで使われていた変数の集合である．

   - 時間遷移は上記の離散遷移のいずれかの後に行う．ロケーションの述語 $b^{l_i}\Psi^{l_i}$，不変条件 $Inv(l_i)$ を用いて時間遷移により ζ_i^{out} に到達可能な凸多面体 ζ_i^{in} を求める．

$$\zeta_i^{in} = Inv(l_i)\wedge b^{l_i}\Psi^{l_i}\wedge\mathrm{Tpre}[l_i,\zeta_i^{out}]$$

ここで，$b^{l_i}\Psi^{l_i}$ は，抽象構造上の状態のロケーション l_i におけるビットベクトル b^{l_i} によって決まる述語集合 Ψ^{l_i} が表す領域を意味する．

3. ζ_i^{in} を次の ζ_{i+1}^{in} として，2. を繰り返して一つ前の状態集合を順次求めていく．

4. 最終的に l_0 まで戻ることができ，ζ_0^{in} に全ての変数が 0 である点が含まれるならば，初期状態に戻ることができ反例であると判定される．途中で凸多面体が空集合になった場合は，偽反例と判定され，精錬の処理を行う．

3.4 精錬

ここでは，精錬について説明する．精錬では，反例解析で偽反例と判定された抽象反例が再び現れないように述語を追加し，抽象状態を分割する．例えば以下のような抽象反例 $\omega_{ce}^{\mathcal{A}}$ の $\sigma_0^{\mathcal{A}} \Rightarrow^{\mathcal{A}} \cdots \Rightarrow^{\mathcal{A}} \sigma_j^{\mathcal{A}} \Rightarrow^{\mathcal{A}} \cdots \Rightarrow^{\mathcal{A}} \sigma_{last}^{\mathcal{A}}$ について，反例解析によって $\sigma_k^{\mathcal{A}} \Rightarrow^{\mathcal{A}} \sigma_{k+1}^{\mathcal{A}}$ という遷移の部分で偽反例になったとする．このとき，$\sigma_{k+1}^{\mathcal{A}}$ の状態には，$\sigma_k^{\mathcal{A}}$ から遷移できる状態集合と遷移できない状態集合が存在すると考えられる．このことから反例解析で凸多面体が空となる直前の $\sigma_{k+1}^{\mathcal{A}}$ の状態を分割することで，同一の反例を消すことができる．

3.5 動的ハイブリッド CEGAR 検証器

本研究では，検証器の実装に Java を用いた．また，状態の計算に必要となる凸多面体上の演算を行うためのライブラリとして，当研究室で開発を行っている LAS(Linear Arithmetic Solver) を用いた．

4 まとめ

本稿では，組込みシステムのモデル検査における状態爆発問題を回避するため，仕様記述言語である動的線形ハイブリッドオートマトンと，その検証手法として動的ハイブリッド CEGAR を提案し，検証機の開発を行った．今後の課題として，検証実験による有用性の確認が挙げられる．

参考文献

[1] R. Alur, C. Courcoubetis, T. A. Henzinger and P. Ho. Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems. *LNCS*, Vol.736, pp.209-229, 1992.

[2] P. C. Attie and N. A. Lynch. Dynamic Input/Output Automata: A Formal Model for Dynamic Systems. *LNCS*, Vol.2154, pp.137-151, 2001.

[3] 南翔太, 瀧内新悟, 瀬古口智, 中居佑輝, 山根智. 動的再構成可能プロセッサのモデル化, 仕様記述とモデル検査. コンピュータソフトウェア, Vol.28(1), pp.190-216, 2011.

[4] E. M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-Guided Abstraction Refinement. *LNCS*, Vol.1855, pp.154-169, 2000

[5] E. M. Clarke, A. Fehnker and Z. Han. Verification of hybrid systems based on counterexample-guided abstraction refinement. *LNCS*, Vol.2619, pp.192-207, 2003

[6] R. Alur and T. Dang. Counter-Example Guided Predicate Abstraction of Hybrid Systems. *LNCS*, Vol.2619, pp.208-223, 2003

[7] Thomas A. Henzinger and Xavier Nicollin and Joseph Sifakis and Sergio Yovine. Symbolic model checking for real-time systems. *Information and Computation* 111(2), pp.193-244, 1994.

[8] B. Boigelot and P. Godefroid. Symbolic Verification of Communication Protocols with Infinite State Spaces using QDDs. *Formal Methods in System Design*, Vol.14, pp.237-255, 1999.

[9] B. Boigelot, P. Godefroid, B. Willems and P. Wolper. The Power of QDDs (Extended Abstract). *LNCS*, Vol.1302, pp.172-186, 1997.