Vol.2010-MPS-77 No.30
2010/3/5

時間オートマトンによるフェースディスプレイの上位設計と形式的検証

田川聖治[†1]　高橋佑輔[†1]　加藤暢[†1]

本稿では，コンピュータを利用した教育支援システムに対して，学生の学習意欲の向上を目的としたフェースディスプレイを開発している．フェースディスプレイとは，教育支援システムに同期して表情アニメーションを実行するものである．はじめに，フェースディスプレイと教育支援システムのモデルを，それぞれ時間オートマトンによって構築している．この際，フェースディスプレイのモデルである時間オートマトンのサイズを小さくするための工夫として，顔画像をショットと呼ぶグループに分割している．つぎに，CTL（Computation Tree Logic）時相理論に基づくモデル検証ツールである Uppaal を用いて，二つの時間オートマトンを並列合成したステム全体の到達可能性，安全性，および，活性を確認している．最後に，フェースディスプレイと教育支援システムを，Java により並行プログラムとして実装している．

High-level Design and Formal Verification of Face Display Using Timed Automata

Kiyoharu Tagawa,[†1] Yusuke Takahashi[†1] and Toru Kato[†1]

In order to motivate students for learning, a face display system is developed for a computer aided learning system. The face display system performs the animation of human's faces synchronizing with the computer aided learning system. First of all, both of the systems are modeled respectively as timed automata. In order to reduce the size of the timed automata modelling the face display system, the images of human's faces are divided into some groups called shots. Then the model checking tool based on a temporal logic known as CTL (Computation Tree Logic), i.e., Uppaal, is employed to verify the reachability, the safety and the liveness properties of the two timed automata that move cooperatively. Finally, the face display system and the computer aided learning system are realized as a concurrent program using Java language.

1. はじめに

表情は重要な非言語コミュニケーションの手段であり，認知科学の分野では古くから研究されている[1)]．また，情報工学の分野においても，表情を含む「顔」に関連する研究は活発に行われている[2)]．ここで，情報工学における「顔」の研究は，認識技術と合成技術に大別される．さらに，顔の合成技術の応用研究として，人と機械のコミュニケーション支援を目的としたヒューマンインタフェースの開発があり，様々な事例が報告されている．

たとえば，Chernoff が考案したフェース法[3)] は，多変数データを表現する顔図形の作成手法である．フェース法では，多変数データの各成分を顔図形の目，口，眉などのパーツの変化に割り当てることで，多変数データを一つの表情として提示する．近年，情報処理技術の進歩に伴って，フェース法にはアニメーションの手法が導入され，原子力発電所など大規模で複雑なシステムの状態を瞬時に把握するために利用されている[4)–6)]．

その他，顔の合成技術の応用例として擬人化エージェントがある．ユーザの作業を代行するソフトウェアをエージェントと呼び，それらの中でコンピュータグラフィックス（CG）などで生成された「顔」を持つものが擬人化エージェントである．擬人化エージェントにおける顔の役割は，ユーザに対して親しみ易さを演出することである．さらに，手話アニメーションにより文章を伝達するシステム[7)] では，手話を行うキャラクターの表情が，感情を具体的に伝えるとともに，文章の内容の理解度を高めるために利用されている．

本稿では，コンピュータを利用した教育支援システムに対して，フェースディスプレイを開発する．フェースディスプレイとは教育支援システムに「顔」を提供するものであり，その状態や学生の操作に応じて表情アニメーションを実行する．また，フェースディスプレイに期待される効果は，教育支援システムを利用する学生の学習意欲の向上である．

一般的に，プログラムの教育では，指導者の人数に比べて学生数が圧倒的に多い．このため，独自の教育支援システムを開発し，授業や演習で利用する試みが幾つかの教育機関で行われている[8)]．教育支援システムを利用することで，学生が作成したプログラムの評価などを自動化し，授業や演習の効率化を図ることができる．一方，学生は自ら教育支援システムを操作し，自主的に学習を進める必要がある．このため，教育支援システムを用いる場合，学生の学習意欲の維持が課題となる．さらに，全入時代における大学のユニバーサル化に

†1 近畿大学理工学部
School of Science and Engineering, Kinki University

伴って，大学の授業でも学習に対する学生の興味を喚起する工夫が求められている[9]．

フェースディスプレイが実行する表情アニメーションでは，擬人化エージェントのように親しみ易さを演出するだけでなく，人間の表情が持つ感情の同調性の機能も考慮している．私たちは，嬉しそうな顔を見ることで喜びの感情が沸き起こり，悲しそうな顔を見ることで気分が沈むことを経験することがある．すなわち，表情には送り手の感情を受け手に伝える機能のほか，送り手と同じ感情を受け手に抱かせる同調性の機能がある[10]．ちなみに，他人の表情を観察するだけで喜怒哀楽の感情が発現する現象には，ミラー・ニューロンが深く関与していることが，脳科学の分野において指摘されている[11]．このため，学生のやる気や興味を引き出すための工夫として，学習内容に応じた適切な表情アニメーションを学生に見せ，人間の意欲の根源である感情を刺激することは有効であると考えられる．

通常，コンピュータによる表情アニメーションは，原理的に複数の顔画像を臨界フリッカ周波数[12] に基づく時間間隔で連続的に表示すれば実現できる．また，各顔画像の表情の記述形式としては，Ekman らが開発した FACS （Facial Action Coding System）における AU （Action Unit）などがある[13]．さらに，表情アニメーションにおける顔画像の時間的な推移の記述形式としては，一連の顔画像を時系列にシナリオとして記録する手法[2] のほか，顔パーツを単位として顔画像の変化を記録する表情譜[14] などが提案されている．

フェースディスプレイでは，教育支援システムの状態や学生の操作に同期させて，顔画像の表示パターンを動的に切り換える必要がある．このため，単純に表情アニメーションのみを実行する場合とは異なり，教育支援システムと分けて独立に設計することができない．また，顔画像は写実的である必要はなく，むしろ誇張された表情の方が効果的であると思われる．これらフェースディスプレイにおける表情アニメーションの特徴を考慮し，顔画像の推移の記述形式として，新たに時間オートマトンを用いることを提案する．時間オートマトン[15] とは，有限個のロケーション[⋆1]にクロックを付加した Büchi オートマトンであり，時間制約のある実時間システムのモデルとして利用されている[16]．フェースディスプレイを時間オートマトンによりモデル化することで，イベントや時間制約による顔画像の遷移規則のほか，各顔画像の表示時間を詳細に指定できる．また，顔画像の表示パターンの全体像を視覚的に把握できるため，表情アニメーションのシナリオの設計も容易となる．

フェースディスプレイのような実時間システムの開発では，適切なモデルを用いた上位設計やモデル検証を行うことで，設計の後戻りが少なくなり，開発後の保守や拡張も容易となる[17]．そこで，フェースディスプレイのみならず，教育支援システムについても時間オートマトンとしてモデルを構築する．さらに，時間オートマトンの代表的なモデル検証ツールである Uppaal[18] を用いて，フェースディスプレイと教育支援システムの二つの時間オートマトンを並列合成したシステム全体に対して，CTL （Computation Tree Logic）時相理論[16] の論理式に基づき，到達可能性や活性のほか，デッドロックフリーなどの安全性について検証する．また，教育支援システムにおける処理時間や顔画像の表示時間を変えて，表情アニメーションが教育支援システムの状態と矛盾しないことを確認する．

ちなみに，著者らの先行研究[19] ではフェースディスプレイのモデル化において，各顔画像を時間オートマトンのロケーションに割り当てていた．しかし，意匠的な表情アニメーションを設計する場合，時間オートマトンの構造が複雑となり，その描画や理解が困難となる．また，顔画像の枚数に比例してロケーションの数が増えるため，モデル検証に要する時間も長くなる．そこで，本稿では，表情アニメーションのシナリオに基づき選定した一連の顔画像をショットとし，各ショットのモデルから時間オートマトンを合成している．

最後に，二つの時間オートマトンを用いて設計したフェースディスプレイと教育支援システムを Java により並行プログラムとして実装する．Java のスレッドを利用することで，時間オートマトンでモデル化された複数の実時間システムの Java による実装は比較的に容易である．このため，Uppaal で記述した時間オートマトンから JML（Java Modeling Language）による Java の仕様記述を自動生成するツールも開発されている[20]．

2. 時間オートマトン

時間オートマトンは 6 項組 $(L, l_0, \Theta, A, E, I)$ で定義される[15]．ここで，L はロケーションの集合，$l_0 \in L$ は初期ロケーション，Θ はクロックの集合，A はアクションの集合，E はロケーション間の遷移の集合であり，クロックに関する制約条件の集合を $B(\Theta)$ とすると，$E \subseteq L \times A \times B(\Theta) \times 2^{\Theta} \times L$ が成り立つ．$I : L \to B(\Theta)$ はロケーションに対してインバリアントと呼ぶ時間の制約条件を割り当てる．また，時間オートマトンは Büchi オートマトンと同様に永遠の動作を記述するため，受理状態に相当するものはない．

時間オートマトンのモデル検証ツールである Uppaal[18] は，スウェーデンの Uppsala 大学とデンマークの Aalborg 大学によって開発された．また，Uppaal はグラフィカルなエディタによる実時間システムのモデリング，シミュレーション，および，CTL 時相論理の論理式に基づくモデル検証が行える総合開発環境でもある．ここで，上記の時間オートマトンを用いれば，抽象化された実時間システムのモデルは構築できる．しかし，フェースディ

⋆1 ロケーション（location）は有限オートマトンの状態に相当する．

Vol.2010-MPS-77 No.30
2010/3/5

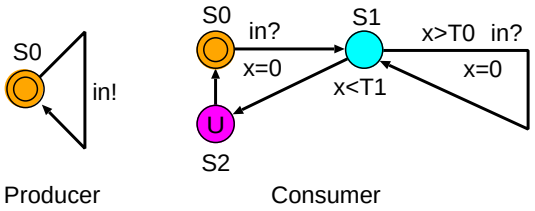

図 1 時間オートマトン
Fig. 1 Timed Automata

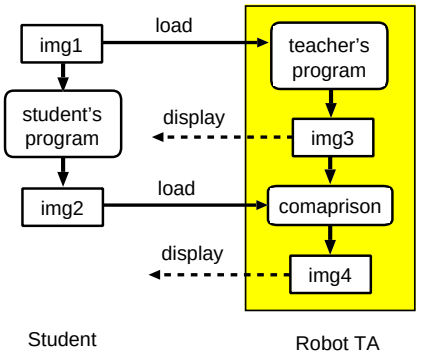

図 2 ロボット TA の機能
Fig. 2 Function of robot TA

図 3 ロボット TA による画像表示
Fig. 3 Dislay of image on robot TA

スプレイのように本格的な実時間システムの振る舞いを正確に記述するためには，6 項組の標準的な時間オートマトンでは表現能力に限界がある．このため，Uppaal では標準的な時間オートマトンに拡張を加えた拡張時間オートマトンをモデル検証の対象とする．

Uppaal が対象とする時間オートマトンでは，クロックのほか，整数の変数や配列が扱える．時間に関する制約条件には，ロケーションに付随するインバリアントのほか，遷移に付随するガードある．また，特別なロケーションとして，時間の経過が許されない Urgent（U）と Committed（C）がある．ここで，後者のロケーション（C）は前者（U）よりも制約が厳しく，ロケーション（C）からの遷移は常に最優先される．さらに，複数の時間オートマトンは，チャネルを介したイベントの送受信により同期を取ることができる．

Uppaal の時間オートマトンの具体例として，図 1 に二つの実時間システム Producer と Consumer のモデルを示す．丸はロケーションを表わし，二重丸は初期ロケーションである．ロケーション間を結ぶ矢印は遷移を示している．ここで，Consumer はクロック `x` を持ち，その時間はロケーション `S0` と `S1` において一様かつ稠密に経過する．また，`S1` に付随する時間制約式 `x<T1` がインバリアント，`S1` からの遷移に付随する時間制約式 `x>T0` がガードである．さらに，Consumer と Producer はチャネルを介してイベント `in` を送受信する．すなわち，Producer が `in!`で送信したイベントを Consumer は `in?`で受信する．

Consumer で遷移が起きる条件は，その遷移に付随するガードと遷移先のインバリアントを満たし，イベント `in` が同期することである．このため，Consumer は `in` を受けて初期ロケーション `S0` から `S1` に遷移する．このとき，クロックを `x=0` とリセットする．その後，クロック `x` の時間が `T0` を経過した後，`T1`(`T1` > `T0`) に達するまでの任意の時刻で Producer とイベント `in` で同期すると，Consumer はクロックを再びリセットして `S1` に戻る．一方，ガードとインバリアントの時間制約式を満たす期間内 (`T0` < `x` < `T1`) にイベント `in` で同期が取られなければ，Consumer は `S1` から瞬時に `S2` を経て `S0` に遷移する．

3. ロボットの TA の設計

3.1 ロボット TA の機能

著者の一人が開発したロボット TA（Teaching Assistant）は教育支援システムの一種であり，画像処理のプログラミング実習において，実際に学生達に使用させている．そこで，本稿では，ロボット TA に「顔」を提供するフェースディスプレイを開発する．

学生は講義で画像処理のアルゴリズムを学んだ後，演習で画像処理プログラムを作成する．その後，学生が作成した画像処理プログラムをロボット TA が点検し，その合否を学生に通知する．高性能なサーバとネットワークを必要とする e-Learning 等とは異なり，ロボット TA はスタンドアロンのソフトウェアである．このため，学生は Web ページからロボット TA をダウンロードして，各人のパソコン上で自由に繰り返し使用できる．

ロボット TA の原理を図 2 のフローチャートに基づき説明する．はじめに，学生は課題の画像処理プログラムを作成した後，与えられたテスト画像を入力画像（img1）として画像処理を施した出力画像（img2）を生成する．つぎに，学生はロボット TA を起動して，入力画像と出力画像のファイルをロボット TA に読み込ませる．ロボット TA は正解の画像処理プログラムを内蔵しており，入力画像を自動的に処理して正解画像（img3）を生成する．さらに，ロボット TA は正解画像と出力画像をピクセルごと比較して，両方の画像が完全に

Vol.2010-MPS-77 No.30
2010/3/5

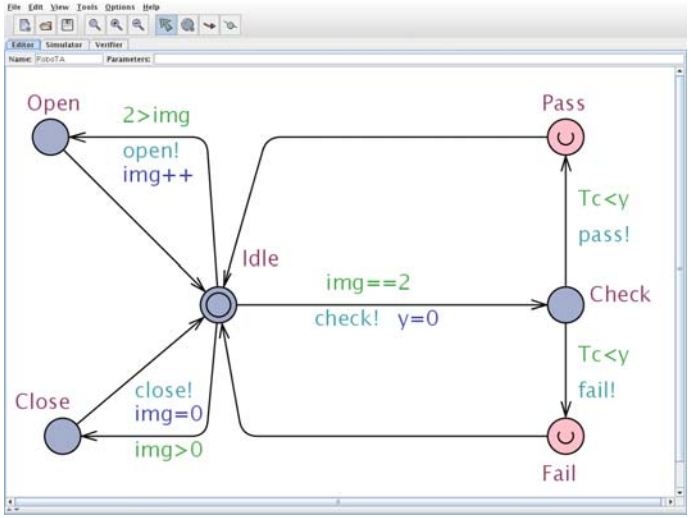

図 4 ロボット TA のモデル（Robot）
Fig. 4 Model of robot TA (Robot)

一致すれば，その旨を文字表示により学生に通知する．一方，出力画像と正解画像が異なる場合，誤り箇所を明示した添削画像（img4）を作成し，正解画像と添削画像をディスプレイ上に並べて表示する．もちろん，正解画像は表示されるだけで，画像ファイルとして取り出すことはできない．図 3 はロボット TA が 2 枚の画像を表示した様子であり，上段には画像ファイルの読み込みボタンと，点検のためのテストボタンがある．

3.2 ロボット TA のモデル

時間オートマトンによりロボット TA のモデルを構築する．まず，Uppaal のグラフィカルなエディタを用いて描いたロボット TA の時間オートマトンを図 4 に示す．ロケーションはアイドル「`Idle`」，画像を開く「`Open`」，画像を閉じる「`Close`」，画像の点検中「`Check`」，合格「`Pass`」，不合格「`Fail`」の 6 個であり，初期ロケーションは `Idle` とする．また，時間オートマトンは，一つのクロック `y` と画像の枚数を表わす変数 `img` を持つ．

まず，学生がロボット TA を起動したとき，図 4 の時間オートマトンは初期ロケーション `Idle` に位置する．つぎに，入力画像（img1）か出力画像（img2）を開くと，イベント

図 5 無表情な瞬き
Fig. 5 Blinking of poker face

図 6 「驚き」と「喜び」
Fig. 6 Surprise and joy

`open!`を発生してロケーション `Idle` から `Open` に遷移するとともに，変数 `img` をインクリメントする．その後，`Open` で画像ファイルを読み込むための時間を費やして `Idle` に戻る．同様に，画像を閉じた場合は，イベント `close!`を発生して `Idle` から `Close` に遷移するとともに，変数 `img` の値を零とする．画像の点検は入出力画像を開いた `img==2` の状態でのみ実行でき，`Idle` からイベント `check!`を発生して `Check` に遷移する．その際，クロックを `y=0` とリセットする．クロック `y` は画像の点検時間を計測するために使用し，点検時間の下限値を `Tc` として，時間制約式 `Tc` < `y` を `Check` からの遷移のガードとしている．`Check` における点検の結果，合格の場合はイベント `pass!`，不合格の場合はイベント `fail!`を発生し，それぞれ `Pass` または `Fail` を経由して初期ロケーションの `Idle` に戻る．

4. フェースディスプレイの設計

4.1 フェースディスプレイの機能

フェースディスプレイは，ロボット TA の状態に応じて表情アニメーションを実行する．顔画像は 6 基本表情[13]（驚き，恐怖，嫌悪，怒り，幸福，悲しみ）と無表情，および，それらの表情を補間する計 28 枚の顔画像を準備した．顔画像のキャラクターとしては，学生達の大半が男性であることから，表情が持つ同調性の機能を期待して，多くの学生が自らを投影しやすい男性とした．また，各顔画像は AU[13] に基づく感情の表情による表現方法を参考にして，市販のグラフィック・ソフトウェア Poser7® によって作成した．

まず，ロボット TA を起動すると無表情な男性の顔が現れる．このとき，図 5 のような目を開いた顔画像と閉じた顔画像の二枚を適当なタイミングで交互に表示することで，無表情な顔に「瞬き」の動作を繰り返させる．無表情な顔も常に動かすことで，不自然さが軽減されるとともに，フェースディスプレイが正常に機能していることが確認できる．

つぎに，フェースディスプレイの表情アニメーションについて，以下のようなシナリオを考えた．すなわち，ロボット TA に入出力画像のファイルを読み込むと，フェースディスプ

Vol.2010-MPS-77 No.30
2010/3/5

f6[0]=61; t6[0]=5;　f6[1]=62; t6[1]=8;　f6[2]=63; t6[2]=5;　f6[3]=64; t6[3]=6;　f6[4]=65; t6[4]=8;

図 7　「怒り」から「悲しみ」のショット
Fig. 7　Shot from anger to sorrow

レイは無表情から期待を込めた「幸福」の表情に変化する．一方，画像を閉じた場合は「嫌悪」となる．また，ロボット TA が画像を点検している最中は，不安な意味での「恐怖」とする．さらに，点検の結果が合格となった場合は，図 6 のように「驚き」から「幸福」に変化する．一方，不合格となった場合，表情は「怒り」から「悲しみ」に変化する．

上記のように，特徴的な表情の顔画像は，ロボット TA の状態の変化に同期させて表示する．その後，フェースディスプレイは，適当な時間制約に従って幾つかの顔画像を経由し，途中でロボット TA が操作されなければ，自動的に無表情な「瞬き」の動作に戻る．

4.2　ショットのモデルと実装

時間オートマトンによる表情アニメーションの記述形式について検討する．まず，表情アニメーションの設計では，ロボット TA の状態や時間の経過に応じて表示する顔画像と順序，および，各顔画像の表示時間を決定すればよい．したがって，時間オートマトンのロケーションに顔画像を対応させれば，自然にフェースディスプレイのモデルが得られる[19]．しかし，意匠的な表情アニメーションを設計する場合，時間オートマトンの構造が複雑となり，その描画や理解が困難となる．また，ロケーションの数が顔画像の枚数に比例するため，顔画像の枚数が多くなると Uppaal によるモデル検証に要する時間も長くなる．

本稿では，上記の問題点を解消するため，連続的に表示する顔画像をショットとすることで，表情アニメーションに含まれる顔画像の集合を構造化する．まず，前述の表情アニメーションのシナリオに沿って，28 枚の顔画像を 6 種類のショットに分割する．ただし，一枚の顔画像は 1 種類以上のショットに属する．たとえば，ロボット TA による画像の点検結果が不合格となった場合，フェースディスプレイの表情アニメーションは図 7 のように「怒り」から「悲しみ」に変化する．図 7 の各顔画像の下に示した数値は，顔画像の識別番号 `f6[k]` と，時間 `Ts` を単位とする表示時間 `t6[k]` である．ここで，図 7 に示したような一連の顔画像を一つのショットとする．ちなみに，図 7 は第 6 番目のショットである．

```
int [] t6 = {61, 62, 63, 64, 65};
int [] f6 = {5, 8, 5, 6, 8};
me.resetEv();
for(int k=0; k<5; k++){
  int z = t6[k];
  this.f = f6[k];
  while(z > 0 && me.getEv() == 0){
    repaint();
    try{
      Thread.sleep(Ts);
    }
    catch(InterruptedException ee){};
    z--;
  }
  if(me.getEv() != 0) break;
}
```

図 8　ショットのプログラム
Fig. 8　Program of shot

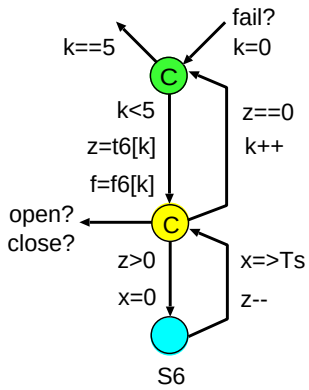

図 9　ショットのモデル
Fig. 9　Model of shot

フェースディスプレイにおいて，図 7 のショットは図 8 のようなプログラムで実装される．図 8 のプログラムでは，`for` 文内において，`k` 番目の顔画像の表示時間 `t6[k]` が変数 `z` に設定され，その顔画像の識別番号 `f6[k]` が変数 `this.f` に設定される．つぎに，`while` 文内において，変数 `this.f` に設定された顔画像がメソッド `repaint()` により表示される．ここで，一枚の顔画像が表示される最少時間は `Ts` である．したがって，変数 `this.f` に設定された顔画像が表示される時間は最長で `z`×`Ts` となる．ただし，途中で `me.getEv()` の結果が非零になると，`break` によって `for` 文から抜け，ショットの処理は終了する．ここで，`me` は後述するイベント・クラスのオブジェクトであり，ロボット TA がイベントの番号を変数 `me.Ev` に書き込む．一方，フェースディスプレイはショットの処理に入る際に `me.resetEv()`

で me.Ev を 0 に初期化するとともに，me.getEv() で me.Ev の値を監視する．

時間オートマトンを用いてショットのモデルを構築する．フェースディスプレイの時間オートマトンにおいて，各ショットを一つのロケーションに割り当てることも考えられる．しかし，表情アニメーションでは各顔画像の表示時間が重要となる．また，フェースディスプレイのモデル検証では個々の顔画像も区別したい．したがって，図 7 のようなショットの振る舞いを記述するモデルでは，顔画像の種類のみならず，それらの表示時間も明示する必要がある．このため，各ショットを一つのロケーションとすることは適切でない．

本稿では，ショットを実装するプログラムに基づき，一つのショットを三つのロケーションから成る時間オートマトンによりモデル化する．たとえば，図 8 に示したショットのプログラムは図 9 の時間オートマトンとなる．まず，図 9 の時間オートマトンでは，ロボット TA からのイベント fail!に fail?で同期し，変数 k を k=0 と初期化して最上段のロケーションに至る．つぎに，ガードが k<5 であるため z=t6[k]，f=f6[k] とし，中段のロケーションに遷移する．最上段と中段のロケーションでは時間は経過しない．さらに，z>0 ならばクロックを x=0 とリセットし，最下段のロケーション S6 に遷移する．S6 は変数 f に設定された顔画像の表示に対応する．S6 で Ts 以上の時間が経過すると，S6 を離れて z--と更新し，中段のロケーションに戻る．ここで，ロボット TA が新たなイベントを発生しない限り，時間オートマトンは S6 を含む二重ループを巡回しながら変数 z と k の値を更新し，やがて最上段のロケーションで k==5 に達するとショットの処理を終了する．

4.3 フェースディスプレイのモデル

各ショットをモデル化した時間オートマトンを結合することで，フェースディスプレイの時間オートマトンによるモデルを構築した．時間オートマトンを用いて表わしたフェースディスプレイ全体のモデルを図 10 に示す．図 10 の時間オートマトンには，前述の表情アニメーションのシナリオに沿って定義された 6 種類のショットが含まれている．

まず，第 1 番目のショットは，無表情な瞬きの動作を繰り返すもので，ロケーション S1 と初期ロケーションを含む．第 2 番目のショットは，入出力画像を開いた際に実行されるもので，ロケーション S2 を含みイベント open に同期して開始される．第 3 番目のショットは，入出力画像を閉じた際に実行されるもので，ロケーション S3 を含みイベント close に同期して開始される．第 4 番目のショットは，画像を点検中に不安な気持ちを表現した表情アニメーションを繰り返すもので，ロケーション S4 を含みイベント check に同期して開始される．第 5 番目のショットは，点検結果が合格の場合に実行される「驚き」から「喜び」への推移であり，ロケーション S5 を含みイベント pass に同期して開始される．最後に，第 6 番目のショットは，点検結果が不合格の場合に実行される「怒り」から「悲しみ」への推移であり，ロケーション S6 を含みイベント fail に同期して開始される．

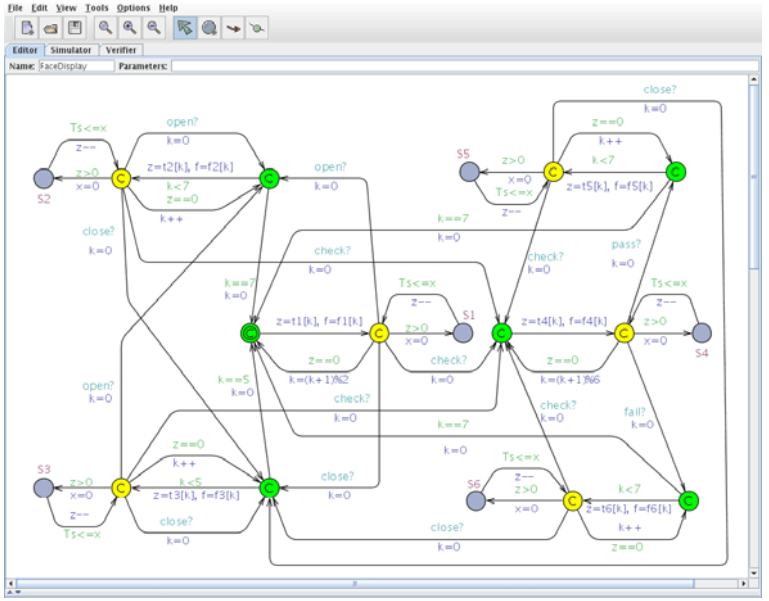

図 10 フェースディスプレイのモデル（Face）
Fig. 10 Model of face display (Face)

5. フェースディスプレイとロボット TA のモデル検証

5.1 Uppaal によるモデル検証

モデル検証ツールである Uppaal によれば，時間オートマトンでモデル化された実時間システムに対して，CTL 時相理論に基づく以下のような論理式を検証できる．

(1) E◇ ϕ
(2) A□ not ϕ
(3) A□ (ω imply (A◇ ϕ))
(3′) ω --> ϕ

論理式 (1) は到達可能性と呼ばれ，いつかは項 ϕ により記述された状況に到達することを検証する．また，論理式 (2) は安全性と呼ばれ，実時間システムがどの様に振る舞っても，項 ϕ によって記述された状況に至らないことを検証する．さらに，論理式 (3) は活性と呼ばれ，項 ω により記述された状況が発生すれば，必ずいつかは状況 ϕ に至ることを検証する．ただし，Uppaal の文法に従えば，論理式 (3) は論理式 (3′) のように記述する．

5.2 検証の内容

フェースディスプレイとロボット TA の時間オートマトンを並列合成した実時間システムのモデルに対して，Uppaal を用いて検証した論理式の一例を以下に紹介する．ただし，フェースディスプレイをモデル化した図 10 の時間オートマトンを「Face」とし，ロボット TA をモデル化した図 4 の時間オートマトンを「Robot」とする．

(a) E◇ `Face.S6`

(b) E◇ (`Robot.y>20` and `Face.f==61`)

(c) A□ not (`Robot.Pass` and `Face.f==61`)

(d) `Robot.Fail --> Face.f==61`

(e) A□ not deadlock

論理式 (a) は到達可能性であり，いつかはフェースディスプレイにおいて第 6 番目のショットが実行されることを検証している．論理式 (b) も到達可能性であり，ロボット TA による画像の点検開始から 20 単位時間が経過した後に，フェースディスプレイが識別番号 61 の「怒り」の顔画像を表示することを検証している．このとき，ロボット TA における画像の点検時間の下限値を `Tc`= 20 とした．論理式 (c) は安全性であり，ロボット TA が合格と判定したときは，フェースディスプレイは「怒り」の顔画像を表示しないことを検証している．また，論理式 (d) は活性であり，ロボット TA が不合格と判定したならば，フェースディスプレイが「怒り」の顔画像を表示することを検証している．

論理式 (e) は安全性の一種であり，フェースディスプレイとロボット TA がデッドロックに陥らないことを検証する．当初，時間オートマトンによるモデルの作成過程では，しばしばデッドロックが発生した．デッドロックのおもな原因は，フェースディスプレイのモデルの不備であった．たとえば，画像の合否判定は，画像ファイルを更新した後に実行されることを想定していたが，実際には学生が画像ファイルを更新せずに合否判定を繰り返し実行することも可能である．上記のようなモデルの不備は，デッドロックが発生するたびに Uppaal が示す反例を解析することで修正した．さらに，顔画像の表示における単位時間 `Ts` と画像の点検時間 `Tc` の値を変えて，`Tc` が非常に長くなった場合でも，実時間システムがデッドロックに陥らないことを確認した．図 10 に示したフェースディスプレイのモデルは，その様なモデルの検証と修正を繰り返して，最終的に得られたものである．

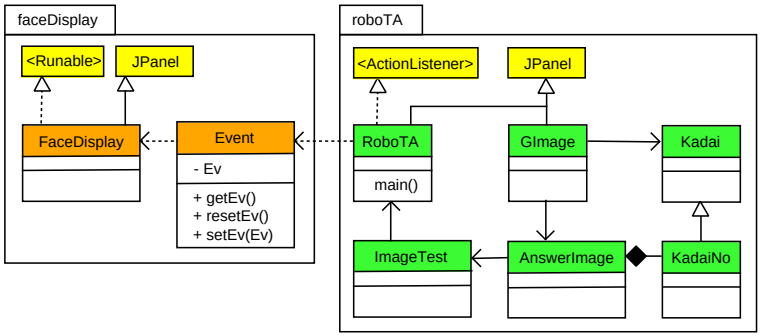

図 11 フェースディスプレイとロボット TA のクラス図
Fig. 11 Class diagram of face display and robot TA

6. フェースディスプレイとロボット TA の実装と評価

フェースディスプレイとロボット TA は Java により並行プログラムとして実装した．その並行プログラムの UML（Unified Modeling Language）によるクラス図を図 11 に示す．既存のロボット TA のプログラムはパッケージ「roboTA」に，新たに開発したフェースディスプレイのプログラムはパッケージ「faceDisplay」にまとめてある．

パッケージ「roboTA」において，「GImage」は画像のクラス，「Kadai」は課題に対する正解プログラムのアブストラクト・クラスであり，「Kadai」を継承して幾つもの正解プログラムのクラス「KadaiNo」が存在する．また，「ImageTest」と「AnswerImage」は画像ファイルの点検と添削に関与するクラスである．クラス「RoboTA」は汎用のクラス「JPanel」を継承するとともに「ActionListener」を実装し，ロボット TA のユーザ・インタフェースを提供する．また，メイン・メソッドはクラス「RoboTA」に存在している．

パッケージ「faceDisplay」において，クラス「FaceDisplay」は図 11 の時間オートマトンに基づきフェースディスプレイを実装しており，そのオブジェクトは「RoboTA」のメイン・メソッドからスレッドとして実行される．「Event」はイベントのクラスであり，ロボット TA が発生したイベントを記憶する変数 `Ev` を保持している．ロボット TA はメソッド

`setEv(Ev)` で `Ev` に値を設定し，フェースディスプレイは `getEv()` で `Ev` の値を読む．また，図 8 に示した通り，フェースディスプレイは `resetEv()` で `Ev` を初期化する．

最後に，学生 11 名を被験者としたアンケート調査により，開発したフェースディスプレイの有効性を評価した．その結果，従来のフェースディスプレイを搭載していないロボット TA と比較し，全員がロボット TA に「顔」があった方が良いと答えた．

7. おわりに

本稿では，学生の学習意欲の向上を目的とし，画像処理プログラミングの教育支援システムであるロボット TA に対して，フェースディスプレイを開発した．フェースディスプレイとロボット TA は，時間オートマトンとして個別にモデル化した後，モデル検証ツールの Uppaal を用いて二つの時間オートマトンを並列合成した実時間システム全体の到達可能性，安全性，活性について確認した．さらに，フェースディスプレイとロボット TA を Java により実装し，Java のスレッドの機能によって両者の協調動作を実現した．

上記のように，フェースディスプレイの開発に先立って，そのモデル化と検証を行ったことで，プログラムを機械的に作成することができた．また，将来のフェースディスプレイの保守や拡張も容易となった．ここで，時間オートマトンを用いたフェースディスプレイの設計方法は，対象とした教育支援システムに依存しない汎用性の高いものである．さらに，顔画像を時間オートマトンの遷移規則に基づき表示する手法は，動画像や本格的な CG アニメーションに比べて，開発コストのほか，コンピュータに与える負荷も小さい．このため，学生のパソコン上で実行させる様々な教育支援システムに応用可能である．

今後の課題として，ゲーム感覚を取り入れた表情アニメーションのシナリオの工夫が挙げられる．たとえば，不合格の回数に応じて顔画像の表示パターンを変えたり，フェース法のように出力画像の誤りの程度によって顔画像を変えることが考えられる．また，フェースディスプレイの有効性の評価では，より大規模な調査と統計的な分析が必要である．

参考文献

1) 吉川左紀子，益谷　真，中村　真：顔と心：顔の心理学入門，サイエンス社 (1993).
2) 長谷川修，森島繁生，金子正秀：「顔」の情報処理，電子情報通信学会論文誌 D-II, Vol.J80-D-II, No.8, pp.2047–2065 (1997).
3) Chernoff, H.: The use of faces to represent points in k-dimensional space graphically, *Journal of the American Statistical Association*, Vol.68, No.342, pp.361–368 (1973).
4) 野口典正，安居院猛，中島正之：表情アニメーションによる多変量データ表示システム，電子情報通信学会論文誌 D， Vol.J70-D, No.11, pp.2177–2181 (1987).
5) Pflughoeft, K.A., Zahedi, F.M. and Soofi, E.: Data visualization using figural animation, *Proceedings of Americas Conference on Information Systems*, pp.1701–1705 (1990).
6) Wyatt, R.: Face charts: A better method for visualizing complicated data, *Proceedings of IADIS International Conference Computer Graphics and Visualization*, pp.51–58 (2008).
7) 児玉哲彦，安村通晃：表情を表現を含む手話アニメーションの試作，情報処理学会研究速報，2003-HI-103， Vol.2003, No.47, pp.23–29 (2003).
8) 島川博光：支援システム導入によるプログラミング教育改善の試み：実授業への適用結果を中心に，システム／制御／情報， Vol.53, No.11, pp.453–458 (2009).
9) 社会法人私立大学情報教育協会：平成 19 年度 私立大学教員の授業改善白書 (2008).
10) 大坊郁夫：しぐさのコミュニケーション：人は親しみをどう伝えあうか，サイエンス社 (1998).
11) Oztop, E., Kawato, M. and Arbib, M.: Mirror neurons and imitation: A computationally guided review, *Neural Networks*, Vol.19, No.3, pp.254–271 (2006).
12) 美濃導彦，西田正吾：情報メディア工学，オーム社 (1999).
13) P. エクマン，W.V. フリーセン,（工藤力訳編）：表情分析入門，誠信書房 (1987).
14) 平山高嗣，川嶋宏彰，西山正紘，松山隆司：表情譜：顔パーツ間のタイミング構造に基づく表情の記述，ヒューマンインタフェース学会論文誌， Vol.9, No.2, pp.201–211 (2007).
15) Alur, R. and Dill, D.L.: A theory of timed automata, *Journal of Theoretical Computer Science*, Vol.126, No.2, pp.183–235 (1994).
16) E.M.Clarke, J., Grumberg, O. and Peled, D.A.: *Model Checking*, The MIT Press (1999).
17) 岡野浩三：上位設計におけるシステムの振る舞い検証技術，システム／制御／情報, Vol.52, No.9, pp.328–333 (2008).
18) Behrmann, G., David, A. and Larsen, K. G.: A tutorial on uppaal, http://www.uppaal.com (2004).
19) 田川聖治，高橋佑輔：時間オートマトンによるフェースディスプレイのモデル化と検証，第 22 回自律分散システム・シンポジウム資料，pp.309–314 (2010).
20) 田辺　誠：モデル検証ツール Uppaal による JAVA 仕様記述支援，第二回システム検証の科学技術シンポジウム予稿集，pp.177–181 (2005).