یک زبان مدل‌سازی مبتنی بر اکتور برای سیستم‌های سایبر-فیزیکی

ایمان جهاندیده

چکیده- یکی از چالش‌های مدل‌سازی سیستم‌های سایبر-فیزیکی[1]، ادغام رفتارهای فیزیکی با رفتارهای محاسباتی و شبکه است و با توجه به این که سیستم‌های سایبر-فیزیکی معمولا دارای مولفه‌ی توزیع شده هستند، مدل‌های مبتنی بر اکتور، چارچوب مناسبی را برای مدل‌سازی چنین سیستم‌هایی فراهم می‌کنند. در این پروژه یک زبان مدل‌سازی صوری مبتنی بر اکتور برای سیستم‌های سایبر-فیزیکی ارائه شده است، که توانایی توصیف رفتارهای فیزیکی، محاسباتی و شبکه را فراهم می‌کند.

کلمات کلیدی- سیستم‌های سایبر-فیزیکی، مدل‌سازی، مدل-سازی مبتنی بر اکتور

۱-مقدمه

سیستم‌های سایبر-فیزیکی از ادغام فرایندهای محاسباتی و فیزیکی ایجاد می‌شوند. در این سیستم‌ها، رایانه‌های نهفته[2] از طریق شبکه، فرآیندهای فیزیکی را کنترل می‌کنند که معمولا همراه با حلقه‌ی بازخورد[3] است، به این معنی که فرآیندهای فیزیکی بر فرآیندهای محاسباتی تاثیر می‌گذارند و بلعکس. به همین خاطر برای تحلیل و توسعه‌ی این سیستم‌ها نیاز به آگاهی از تعامل و اشتراک بین سیستم رایانه‌ای، نرم‌افزار، شبکه و فرایندهای فیزیکی است.

سیستم‌های سایبر-فیزیکی چند تفاوت کلیدی با سیستم‌های نرم-افزاری دارند. در سیستم‌های نرم‌افزاری زمان اجرای یک دستور تنها مرتبط با کارایی سیستم است و نه درستی آن ولی در مقابل در سیستم‌های سایبر-فیزیکی زمان اجرای یک دستور می‌تواند برای درستی سیستم حیاتی باشد. همچنین فرآیندهای فیزیکی ترکیب رخداد چندین رویداد همزمان هستند، بر خلاف فرایندهای نرم‌افزاری که به صورت گام‌های متوالی اجرا می‌شوند.

با توجه به کاربرد سیستم‌های سایبر-فیزیکی در زمینه‌های ایمنی-مهم[4] ،ارزیابی و بررسی این سیستم‌ها به شدت حیاتی است. روش‌های مبتنی بر مدل، یکی از روش‌های اصلی در طراحی چنین سیستم‌هایی است. این مدل‌ها علاوه بر تعیین مشخصات سیستم، توانایی شبیه‌سازی و بررسی آن را فراهم می‌کنند.

یکی از پایه‌ای‌ترین زبان‌های مدل‌سازی برای سیستم‌های سایبر-فیزیکی، خودکاره‌های ترکیبی[5] است. این مدل می‌تواند رفتارهای پیوسته (فیزیکی) و رفتارهای گسسته (سایبری) را توصیف کند. به طور خلاصه یک خودکاره‌ی ترکیبی دارای چندین مکان برای مدل‌کردن حالات گسسته سیستم و دارای چندین متغیر پیوسته و معادلات زمانی آن‌ها برای بیان رفتار پیوسته‌ی سیستم است. با این همه مدل سازی رفتارهای پیچیده سایبری و مدل‌سازی رفتارهای شبکه در این زبان سخت است.

در مقابل Timed Rebeca [1] یک زبان برای مدل‌سازی سیستم‌های غیرسنکرون بی‌درنگ گسسته است. این زبان مبتنی بر اکتور است و دارای مفاهیمی همچون زمان پردازش، ارسال و دریافت پیام به صورت غیرسنکرون، تاخیر شبکه و مهلت برداشتن پیام است. با این حال این زبان برای مدل‌سازی رفتار پیوسته، مفهومی ارائه نمی‌دهد.

در این پروژه زبان مبتنی بر اکتور HPalang ارائه شده است که به طور خلاصه ویژگی‌های زیر را دارد:

- مدل معنایی صوری مبتنی بر خودکاره‌ی ترکیبی
- مدل‌سازی ارسال و دریافت پیام به صورت غیرسنکرون
- مدل‌سازی زمان پردازشی بخش سایبر
- مدل‌سازی رفتارهای فیزیکی و پیوسته

۱-۱- تعاریف اولیه

در این بخش بعضی از مفاهیمی که در ادامه وجود دارند، تعریف شده-اند.

۱-۱-۱- سیستم گذار[6]

---

[1] Cyber-Physical

[2] Embedded

[3] Feedback Loop

[4] Safety-Critical

[5] Hybrid Automata

[6] Transition System

یک سیستم گذار یک چندتایی به صروت $(S, \rightarrow, I, L, s_0)$ است که:

- S مجموعه‌ی حالات است.
- L مجموعه‌ی برچسب‌ها است.
- $\rightarrow \subseteq S \times L \times S$ رابطه‌ی گذارها است.
- s_0 حالت شروع است.

۱-۱-۲- خودکاره‌ی ترکیبی

یک خودکاره‌ی ترکیبی [1] به صورت یک چندتایی $(Loc, V, (l_0, v_0), \rightarrow, I, F)$ تعریف می‌شود که:

- Loc مجموعه محدود مکان‌ها است.
- V مجموعه‌ای از متغییرهای پیوسته است.
- l_0 مکان شروع و v_0 مقادیر اولیه متغییرهای V است.
- $\rightarrow \subseteq Loc \times B(V) \times Reset(V) \times Loc$ رابطه‌ی پرش‌ها است که:
  - $B(V) \subseteq Val(V)$ شروط ارضا شدن پرش و
  - $Reset(V)$ مجموعه‌ای از مقداردهی به متغییرهای V پس از پرش است.
- $I : Loc \rightarrow B(V)$ مقادیر مجاز متغییرها را در هر مکان مشخص می‌کند.
- $F : Loc \rightarrow B(V \cup \dot{V})$ محدودیت‌ها روی متغییرها و مشتق آن‌ها توصیف می‌کند و رفتار پیوسته در هر مکان را مشخص می‌کند.

۱-۱-۳- قوانین *SOS*

این قوانین از دو بخش فرض و نتیجه تشکیل شده‌اند که در شکل ۱ ساختار نمایشی این قانون آورده شده است. این قانون به معنی این است که در صورت برقرار بودن فرض، نتیجه نیز برقرار است.

$$\frac{\text{premise}}{\text{conclusion}}$$

شکل ۱: ساختار قانون SOS

۲- مسئله و کارهای مشابه

زبان‌های مختلفی برای مدل‌سازی و تحلیل و درستی‌یابی سیستم‌های سایبر-فیزیکی ارائه شده است. این زبان‌ها در صوری بودن مدل معنایی و نوع تحلیل و ساختار مولفه‌ها متفاوت‌اند.

Ptolemy [2] یک چارچوب مدل‌سازی، شبیه‌سازی و طراحی سیستم‌ها بی‌درنگ و هم‌روند مبتنی بر اکتور است. تمرکز این چارچوب بر ادغام مدل‌های محاسباتی ناهمگون است. مدل محاسباتی مجموعه قوانین مربوط بر اجرای همروند مولفه‌ها و طریقه‌ی ارتباط بین آن‌ها است. در Ptolemy مدل‌های محاسباتی مختلفی از جمله ماشین حالت متناهی[۷]، زمان پیوسته، واکنشی-سکنرون[۸]، گسسته رخداد[۹] و جریان داده‌ای[۱۰] گنجانده شده است. برای ایجاد مدل‌های ناهمگون، از اکتورهای سلسله مراتبی استفاده می‌شود که هر اکتور مرکب شامل یک کارگردان[۱۱] و چندین اکتور دیگر است. کارگردان در مدل، درواقع همان مفهوم مدل محاسباتی است که با تغییر کارگردان می‌تواند رفتار مدل را تغییر داد. با توجه به تمرکز این چارچوب بر شبیه‌سازی، مدل معنایی صوری برای Ptolemy تعریف نشده است.

CIF [3] یک زبان مدل‌سازی برای سیستم‌های ترکیبی است. یک مدل در این زبان دارای چند خودکاره و گروه است. خودکاره‌ها رفتار یک سیستم را تعریف می‌کنند و گروه‌ها مجموعه‌ای از خودکاره‌ها و اعلان‌ها هستند. گرامر این زبان نزدیک به خودکاره‌های ترکیبی است. یک خودکاره‌ی ترکیبی در این زبان از تعریف چند location و edgeهای مربوط به آن، توصیف می‌گردد.

زبان Acumen [4] یک زبان مدل‌سازی و شبیه سازی برای سیستم-های ترکیبی است. برای ساخت مدل در این زبان، از مفهوم "مدل" استفاده می‌شود. هر مدل در این زبان حداقل دارای یک تعریف مدل Main است که شروع مدل را مشخص می‌کند و تمام بخش‌های دیگر سیستم باید در این قسمت تعریف یا ساخته شوند. رفتارهای فیزیکی هر مدل در بلاک always توصیف می‌شوند. در این بلاک می‌تواند از دستورات شرطی برای

---

[۷] Finite Transition System

[۸] Synchronous-Reactive

[۹] Discrete Event

[۱۰] Data Flow

[۱۱] Director

تعیین مدل‌های مختلف سیستم استفاده کرد. رفتارهای فیزکی به صورت مشتق متغییرها تعریف می‌شوند.

ابزار SpaceEx [5] یک ابزار درستی‌یابی برای سیستم‌های ترکیبی است. هدف در این ابزار اثبات یک ویژگی ایمنی در مدل سیستم است. SpaceEx دارای سه بخش ویرایشگر مدل برای ویرایش مدل سیستم به صورت بصری، هسته‌ی آنالیز برای بررسی مدل با توجه به پارامترهای ورودی و رابط وب، یک رابط گرافیکی برای استفاده آسان از هسته‌ی آنالیز و مشاهده نتیجه خروجی است. مدل پایه‌ی این ابزار خودکاره‌ی ترکیبی است و برای ماژولار کردن مدل، مفاهیم مولفه‌ی پایه و مولفه‌ی شبکه در این ابزار وجود دارد. مولفه‌ی پایه در واقع یک خودکاره‌ی ترکیبی است و مولفه‌ی شبکه، از اتصال چند مولفه (پایه یا شبکه) ایجاد می‌شود. ارتباط بین مولفه‌ها به صورت برچسب‌های همگام‌سازی روی گذارهای مولفه‌ی پایه بیان می‌شود. برای درستی‌یابی مدل چند الگوریتم دسترسی‌یابی در ابزار تعبیه شده است.

در طراحی زبان HPalang مدل‌سازی بخش‌های مختلف سیستم‌های سایبر–فیزیکی و ویرایش و بهبود کم هزینه‌ی مدل تمرکز اصلی بوده است. مدل‌سازی مبتنی بر اکتور یکی از روش‌های موفق در زمینه‌ی مدل‌سازی سیستم‌ها در انجمن مهندسی نرم‌افزار است. انگیزه‌ی این زبان، ارائه‌ی زبان مدل‌سازی مناسب مبتنی بر اکتور برای سیستم‌های سایبر–فیزیکی است. همچنین با توجه به ساختار مبتنی بر اکتور، این زبان یک روش طراحی پیمانه‌ای را ارائه می‌دهد. این زبان اجازه تعریف رفتارهای سایبری به صورت دستوری فراهم می‌کند که یکی از روش‌های معمول برای بیان رفتار-های سیستمی است.

هرچند روش‌های درستی‌یابی در زمینه‌ی سیستم‌های سایبر–فیزیکی در ابتدای راه هستند ولی روش‌های تقریبی و ایمن برای درستی‌یابی زیرمجموعه‌ای از خودکاره‌های ترکیبی وجود دارد. مدل معنایی زبان HPalang مبتنی بر خودکاره‌های ترکیبی بوده و علاوه بر شبیه‌سازی، اجازه-ی درستی‌یابی مدل‌های این زبان را ارائه می‌دهد.

۳- تعریف زبان

۳-۱- نشانه‌گذاری‌ها

در تعریف زبان HPalang دنباله خالی با نماد ϵ نمایش داده شده است و $\langle h|T\rangle$ دنباله‌ای است که المان ابتدا آن $h \in A$ و $T \in A^*$ دنباله‌ی باقی مانده است که A یک مجموعه است. برای دو دنباله‌ی σ و σ' تعریف شده روی A ، $\sigma \oplus \sigma'$ دنباله‌ی حاصل از اضافه کردن σ' به انتهای σ است.

برای تابع $f: X \to Y$ از نماد $f[x \to y]$ به معنی تابع $\{(a,b) \in f | a \neq x\} \cup \{(x,y)\}$ استفاده شده است.

۳-۲- نحو[۱۲]

مدل HPalang شامل تعریف چند اکتور و یک بلوک main برای پیام-های اولیه است. در شکل ۲ گرامر زبان آمده است.

۳-۲-۱- اکتور

یک اکتور به صورت یک چندتایی به فرم $(\text{id}, \text{vars}, \text{mthds})$ تعریف می شود که id شناسه، vars مجموعه‌ای متغییرهای گسسته و پیوسته و mtds مجموعه متدهای اکتور است. هر متد به صورت یک چندتایی $(m,b) \in MName \times Stat^*$ تعریف می‌شود که m نام متد و b بدنه-ی متد شامل دنباله‌ای از دستورات است. پارامترهای ورودی متد در این نسخه در نظر گرفته نشده است.

۳-۲-۲- دستورات

در این قسمت دستورات این زبان تعریف شده‌اند. $DVar$ و $CVar$ به ترتیب مجموعه‌ی متغییرهای گسسته و پیوسته هستند و $DExpr$ و $CExpr$ به ترتیب عبارات ریاضی شامل متغییرهای گسسته و عبارات ریاضی شامل متغییرهای پیوسته است. وجود پیشوند B قبل از $Expr$ به معنی عبارات با مقادیر بولی است.

$DAssignment = DVar \times DExpr$ عملیات انتصاب یک مقدار گسسته به یک متغییر گسسته است. از نماد $var = expr$ برای نمایش استفاده شده است.

$CAssignment = CVar \times CExpr$ عملیات انتصاب یک مقدار پیوسته به یک متغییر پیوسته است. از نماد $var = expr$ برای نمایش استفاده شده است.

$Cond = DBExpr \times Stat^* \times Stat^*$ دستور شرطی است که دارای یک عبارت بولی گسسته و دو دنباله دستور است. در صورت ارضا شدن شرط، دنباله اول و در غیر این صورت دنباله دوم اجرا می‌شود. از نماد $if\ expr\ then\ \sigma\ esle\ \sigma'$ برای نمایش استفاده شده است.

$Send = ID \cup \{self\} \times MName$ دستور ارسال است که یک پیام به یک اکتور است. از نماد $x.m()$ برای نمایش استفاده شده است.

$Continuous\ Behavior = CBExpr \times (CVar' \times CExpr)^* \times CBExpr \times Stat^*$ دستور یک رفتار پیوسته است. یک رفتار پیوسته‌ی $(inv, ode, guard, actions)$ همانند خودکاره‌ی ترکیبی تعریف شده است. برای سادگی تنها یک گذار در نظر گرفته شده

---

[۱۲] Syntax

است. $actions$ دنباله‌ی دستوراتی است که پس از انجام گذار اجرا می‌شوند. از نماد $inv\ ode\ guard\ actions$ برای نمایش استفاده شده است.

$Delay = R^+$ دستوری است اکتور را برای R^+ واحد زمانی مشغول نگه می‌دارد. از نماد $delay(r)$ برای نمایش استفاده شده است.

$Resume$ دستور ادامه‌ی اجرای دستورات است که پس از پایان $Delay$ اعمال می‌گردد. از نماد $resume()$ برای نمایش استفاده شده است.

```
<model> ::= <actor>* <main>
<actor> ::= 'actor' <actor-id> '{'
        (<state-var> | <method>)* '}'
<state-var > ::= <var-decl > ';'
<var-decl > ::= <type> <var>
<method> ::= <message> '(' ')' '{' <stat-list> '}'
<stat-list> ::= (<statement> ';')*
<statement> ::= <nonblock-stat> | <delay>
<delay> ::= 'delay' '(' expr ')'
<nonblock-stat> ::= <c-assignment> |<d-assignment>|
        <conditional> | <send> | <c-behavior>
<d-assignment> ::= <var> '=' <dexpr>
<c-assignment> ::= <var> '=' <cexpr>
<conditional > ::= 'if' '(' <dexpr > ')'
        <stat-list> 'else' <stat-list>
<send> ::= <actor-id> '.' <message> '(' ')'
<c-behavior> ::= 'inv' '('<cexpr>')' '{'<ode-list>'}
        'guard' '(' <cexpr> ')' '{' <stat-list> '}'
<ode-list> ::= (var ''' )*
<main> ::= 'main' '{' (<send> ';')* '}'
<message> ::= <identifier>
<actor-id> ::= <identifier>
<var > ::= <identifier>
<type> ::= 'int' | 'real'
```

شکل ۲: گرامر زبان HPalang به فرم EBNF

۳-۲- نحو ایستا

قوانین زیر، قوانین مربوط به مناسب بودن ساختار یک مدل در این زبان هستند که به سادگی در گرامر زبان قابل تعریف نیستند ولی به صورت ایستا قابل بررسی هستند.

- اکتورها دارای شناسه‌های یکتا هستند.
- نام متغییرها در یک اکتور یکتا است.
- نام متدها در یک اکتور یکتا است.
- مدل نوع-ایمن است. یعنی:
  - عبارات نوع-ایمن هستند.
  - هر دو طرف یک انتصاب هم نوع هستند.
  - گیرنده‌ی دارای متدی با نام پیغام ارسال شده است.

۳-۳- معنای اجرای

در این بخش معنای صوری این زبان مبتنی بر خودکاره‌ی ترکیبی بیان می‌شود. تعریف این معنا در دو مرحله صورت می‌گیرد. در مرحله‌ی اول یک سیستم گذار تعریف می‌شود و در مرحله‌ی دوم این سیستم‌گذار به یک خودکاره‌ی ترکیبی تبدیل می‌شود.

۳-۳-۱- توابع کمکی

از توابع کمکی زیر را برای تعریف معنایی صوری در بخش‌های بعد، استفاده شده است.

$body : ID \times MName \to Stat^*$ ، که $body(x, m)$ بدنه‌ی متد m از اکتور با شناسه‌ی x را برمی‌گرداند.

$delayVar : ID \to CVar$ که $delayVar(x)$ متغییر پیوسته‌ی تخصیص داده شده برای تاخیر اکتور از اکتور x را برمی‌گرداند.

۳-۳-۲- حالات

اکتورها توسط ارسال پیام ارتباط برقرار می‌کنند و پیام‌های دریافتی را در صف ذخیره می‌کنند. نوع یک پیام به صورت $Msg = MName$ تعریف می‌شود. اکتور دارای دو صف پیام q_h و q_l است که به ترتیب صف مربوط به پیام‌های با اولیت بالا و صف مربوط به پیام‌های با اولیت پایین است. نوع این دو صف به صورت $Queue = Msg^*$ تعریف می‌شود.

حالات سرارسی یک مدل به صورت یک تابع $l : ID \to DS \times CB$ نمایش داده می‌شود که $DS = (Var \to Val) \times Queue \times Queue \times Stat^*$ و $CB = CBExpr \times ODE^* \times CBExpr \times Stat^*$. تابع l شناسه‌ی یک اکتور را به حالت محلی آن نگاشت می‌کند. حالت محلی یک اکتور یک چندتایی به صورت $(v, q_h, q_l, \sigma, cb)$ است که v مقداردهی به متغییرهای گسسته‌ی اکتور، q_h و q_l صف پیام‌ها، σ دنباله‌ی دستورات اجرایی و cb دنباله‌ای از رفتارهای پیوسته‌ی اکتور است.

۳-۳-۳- گذار

در این قسمت گذارهای بین حالات به صورت SOS تعریف شده‌اند.

Low Priority Message Take

$$\frac{\begin{array}{c} l(x) = (ds, cs) \ \wedge\ ds = (v, \varepsilon, m|T, \varepsilon) \wedge \neg v(suspended) \wedge \\ l'(x) = (ds', cs') \ \wedge\ ds' = (v', q_h', q_l', \varepsilon) \ \wedge \\ \forall i \in ID\backslash\{x\}\big(l(i) = (v'', q_h'', q_l'', \varepsilon, cs) \wedge l'(i) = (v'', q_h^*, q_l^*, \varepsilon, cs^*)\big) \wedge \\ l[x[ds] \mapsto (v, \varepsilon, T, body(m))] \Rightarrow^* l' \end{array}}{l \xrightarrow{\tau} l'}$$

High Priority Message Take

$$\frac{\begin{array}{c} l(x) = (ds, cs) \ \wedge\ ds = (v, m|T, q_l, \varepsilon) \ \wedge \\ l'(x) = (ds', cs') \ \wedge\ ds' = (v', q_h', q_l', \varepsilon) \ \wedge \\ \forall i \in ID\backslash\{x\}\big(l(i) = (v'', q_h'', q_l'', \varepsilon, cs) \wedge l'(i) = (v'', q_h^*, q_l^*, \varepsilon, cs^*)\big) \wedge \\ l[x[ds] \mapsto (v, T, q_l, body(m))] \Rightarrow^* l' \end{array}}{l \xrightarrow{\tau} l'}$$

Continuous Behavior Expiration

$$\frac{l(x) = (ds, cs) \ \wedge \ ds = (v, q_h, q_l, \sigma) \ \wedge \ (i, o, g, a) \in hs}{\begin{gathered} l \xrightarrow{g} l[x \mapsto (ds', hs')] \\ ds' = (v, q_h \oplus m, q_l, \sigma) \\ cs' = cs \backslash (i, o, g, a) \\ m = a\ message\ with\ body\ of\ "a" \end{gathered}}$$

Continuous Variable Assignment

$$\frac{l(x) = (ds, cs) \ \wedge \ ds = (v, q_h, q_l, (cvar := expr|\sigma))}{\begin{gathered} l \xrightarrow{\tau, cvar := eval(expr)} l[x \mapsto (ds', cs)] \\ ds' = (v, q_h, q_l, \sigma) \end{gathered}}$$

Discrete Variable Assignment

$$\frac{l(x) = (ds, cs) \ \wedge \ ds = (v, q_h, q_l, (dvar := expr|\sigma))}{\begin{gathered} l \overset{\tau}{\Rightarrow} l[x \mapsto (ds', cs)] \\ ds' = (v[dvar \mapsto eval(expr)], q_h, q_l, \sigma) \end{gathered}}$$

Conitional True

$$\frac{\begin{gathered} l(x) = (ds, cs) \ \wedge \ ds = (v, q_h, q_l, (if\ expr\ \sigma\ else\ \sigma'|\sigma'')) \\ eval(expr) = True \end{gathered}}{\begin{gathered} l \overset{\tau}{\Rightarrow} l[x \mapsto (ds', cs)] \\ ds' = (v, q_h, q_l, \sigma \oplus \sigma'') \end{gathered}}$$

Conitional False

$$\frac{\begin{gathered} l(x) = (ds, cs) \ \wedge \ ds = (v, q_h, q_l, (if\ expr\ \sigma\ else\ \sigma'|\sigma'')) \\ eval(expr) = False \end{gathered}}{\begin{gathered} l \overset{\tau}{\Rightarrow} l[x \mapsto (ds', cs)] \\ ds' = (v, q_h, q_l, \sigma' \oplus \sigma'') \end{gathered}}$$

Resume Statement

$$\frac{l(x) = (ds, cs) \ \wedge \ ds = (v, q_h, q_l, resume|\sigma)}{\begin{gathered} l \overset{\tau}{\Rightarrow} l[x \mapsto (ds', cs)] \\ ds' = (v[suspended := false], q_h, q_l, \sigma \oplus v(comd)) \end{gathered}}$$

Continuous Behavior Statement

$$\frac{\begin{gathered} l(x) = (ds, cs) \\ \wedge \ ds = (v, q_h, q_l, (inv\ expr\ ode\ guard\ expr'\ \sigma|\sigma')) \end{gathered}}{\begin{gathered} l \overset{\tau}{\Rightarrow} l[x \mapsto (ds', cs')] \\ ds' = (v, q_h, q_l, \sigma') \\ cs' = cs \cup (expr, ode, expr', \sigma) \end{gathered}}$$

Delay Statement

$$\frac{\begin{gathered} l(x) = (ds, cs) \\ \wedge \ ds = (v, q_h, q_l, delay(d)|\sigma) \end{gathered}}{\begin{gathered} l \xrightarrow{\tau, D(x) := 0} l[x \mapsto (ds', cs')] \\ ds' = (v[suspended := true][comd := \sigma], q_h, q_l, \varepsilon) \\ cs' = cs \cup (D(x) \leq d, \{D(x)' = 1\}, D(x) == d, \{resume\}) \end{gathered}}$$

Message Send

$$\frac{\begin{gathered} l(x) = (ds_x, cs_x) \\ l(y) = (ds_y, cs_y) \\ ds_x = (v^x, q_h^x, q_l^x, (y, m)|\sigma^x) \\ ds_y = (v^y, q_h^y, q_l^y, \sigma^y) \end{gathered}}{\begin{gathered} l \overset{\tau}{\Rightarrow} l[x \mapsto (ds_x' cs_x)][y \mapsto (ds_y', cs_y)] \\ ds_x' = (v^x, q_h^x, q_l^x, \sigma^x) \\ ds_y' = (v^y, q_h^y, q_l^y \oplus m, \sigma^y) \end{gathered}}$$

۳-۳-۴- سیستم گذار میانی

سیستم گذار میانی یک مدل M به صورت $TS(M) = (l, \rightarrow, s_0, L)$ تعریف می‌شود که:

- l مجموعه حالات سراسری است (مجموعه‌ی تمام توابع از شناسه‌ی اکتورها به حالات محلی)
- $\rightarrow$ کوچکترین رابطه‌ای است که توسط قوانین SOS بالا تعریف شده است
- s_0 حالات اولیه است که متغییرهای گسسته مقدارهی اولیه شده‌اند و پیغام‌های درون بلوک main در صف اولیت پایین اکتورهای مشخص شده گذاشته شده است.
- L برچسب‌هایی است که مطابق قوانین بالا مشخص شده است.

۳-۳-۵- خودکاره‌ی ترکیبی

برای تبدیل $TS(M) = (l, \rightarrow, s_0, L)$ به یک خودکاره‌ی ابتدا دو عمل زیر روی سیستم گذار اعمال می‌شود.

- اولیت‌دهی به گذارهای τ: اگر در حالتی حداقل یک گذار τ وجود داشته باشد، گذارهایی شرط‌دار این حالت حذف می‌شوند.
- ادغام و حذف گذارهای τ: گذارهای τ ادغام شده تا در سیستم گذار نهایی تنها گذارهای شرط دار باقی بمانند.

خودکاره‌ی ترکیبی $HA(Loc, V, (loc_0, v_0), \Rightarrow, I, F)$ حاصل از سیستم گذار به دست آمده به صورت زیر تعریف می‌شود.

- $Loc = l$
- $\Rightarrow$کوچکترین رابطه‌ای است که توسط قانون SOS زیر تعریف شده است

- V مجموعه‌ی متغییرهای پیوسته‌ی مدل
- $loc_0 = s_0$
- v_0 مقداردهی اولیه به متغییرهای پیوسته
- I و F مطابق با قانون SOS زیر مشخص شده است.

$$\frac{l \xrightarrow{g,r} l'}{l \xRightarrow{g,r} l'}$$

$$I(l) = logical\ and\ of\ all\ invariants\ in\ l$$
$$F(l) = set\ of\ all\ odes\ in\ l$$

۴- مثال‌ها

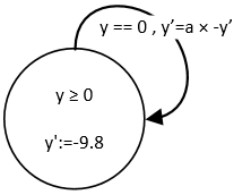
۴-۱- سقوط توپ

در این بخش سقوط آزاد یک توپ مدل می‌شود. در این سیستم، توپ از ارتفاع اولیه رها شده و با شتاب g – در جهت عمودی حرکت می‌کند. زمانی که ارتفاع توپ از سطح زمین صفر شود، سرعت توپ برعکس شده و با ضریب کاهش می‌یابد. این کاهش سرعت، از دست رفتن قسمتی از انرژی توپ با برخود به زمین را بیان می‌کند.

```
actor Ball
{
    real y = h;
    Fall()
    {
        inv(y >=0)
            y'' = -9.8;
        guard(y == 0)
        {
            y' = a * y';
            self.Fall();
        }
    }
}
```

شکل ۳: مدل توپ

مدل تعریف شده این سیستم در شکل ۳ آمده است. در این مدل یک اکتور Ball وجود دارد که رفتار توپ را مدل می‌کند. این اکتور دارای یک متغییر پیوسته‌ی y است که ارتفاع توپ از سطح زمین است و مقدار اولیه آن h است. متد Fall، سقوط توپ را مدل می‌کند. در این متد تنها رفتار پیوسته‌ی سقوط تعریف شده است. در این رفتار تا زمانی که ارتفاع بیشتر از صفر است، مشتق دوم y با نرخ 9.8 – تغییر می‌کند. زمانی که y مساوی صفر شود، مشتق اول y، که سرعت توپ است، عکس شده و با ضریف a کاهش می‌یابد و یک پیام Fall به خود توپ ارسال می‌شود تا عمل سقوط دوباره صورت گیرد. در بلوک main این مدل نیز پیام Fall به اکتور Ball ارسال می‌شود تا توپ شروع به سقوط کند. ترجمه‌ی این مدل به خودکاره ترکیبی در شکل ۴ آمده است.



شکل ۴: خودکاره‌ی ترکیبی مدل توپ

۴-۲- ماشین نوشیدنی

در این سیستم، دستگاه دارای دو نوع نوشیدنی چای و قهوه است. با توجه به درخواست کاربر، دستگاه نوشیدنی مورد نظر را ابتدا تا دمای مشخص شده گرم می‌کند و سپس لیوان را با نوشیدنی انتخاب شده پر می‌کند. دما و حجم نوشیدنی چای به ترتیب ۱۰۰ درجه‌ی سانتی گراد و ۳۰۰ سی سی و دما و حجم نوشیدنی قهوه به ترتیب ۹۰ درجه‌ی سانتی‌گراد و ۲۰۰ سی سی است.

```
actor Machine
{
    int orderType;

    PrepareTea()
    {
        orderType = 0;
        Heater.HeatUp100();
    }
    PrepareCoffee()
    {
        orderType = 1;
        Heater.HeatUp90();
    }

    Heated()
    {
        if(orderType == 0)
            Filler.Fill300();
        else
            Filler.Fill200();
    }
    Filled()
    {
        User.RecieveOrder();
    }
}
```

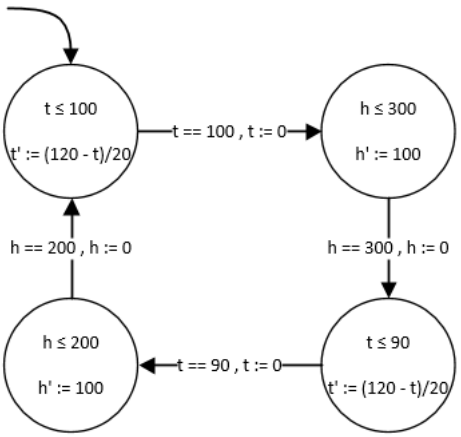
شکل ۵: مدل اکتور Machine

در این مدل چهار اکتور User، Machine، Heater و Filler تعریف شده است. اکتور User رفتار کاربر را مدل می‌کند که در این مثال خواص، کاربر نوشیدنی خود را به طور یکی در میان، تغییر می‌دهد. User با ارسال پیام PrepareTea یا PrepareCoffee به Machine نوع نوشیدنی درخواستی خود را مشخص می‌کند. Machine با دریافت یکی از این دو پیام، ابتدا نوع نوشیدنی را در متغییر گسسته‌ی orderType ذخیره می‌کند و سپس پیام HeatUp100 یا HeatUp90 را با توجه به نوع درخواست به Heater ارسال می‌کند. Heater با دریافت یکی از این دو پیام، یک رفتار پیوسته برای گرم شدن تعریف می‌کند که در این رفتار پس از رسیدن به دمای

مطلوب، متغییر t صفر می‌شود و پیام Heated به Machine فرستاده می‌شود. در متد Heated، Machine با توجه به نوع نوشیدنی درخواستی، پیام Fill200 یا Fill300 را به Filler ارسال می‌کند. Filler با دریافت یکی از این دو پیام، یک رفتار پیوسته برای مدل کردن پرشدن لیوان تعریف می‌کند. در این رفتار پس از رسیدن به حجم مناسب، پیام Filled به Machine ارسال می‌شود. در متد Filled، Machine پیام RecieveOrder را به User می‌فرستاد و User در این متد دوباره یک Order جدید می‌دهد. دلیل استفاده از دو متد در اکتورهای Filler و Heater، عدم تعریف شدن پارامتر ورودی برای متدها، در این نسخه است. در اشکل ۵و ۶مدل اکتور Machine و Heater آمده است. همچنین در شکل ۷ خودکاره‌ی ترکیبی معادل مدل نشان داده شده است.

```
actor Heater
{
    real t;

    HeatUp100()
    {
        inv (t<= 100)
            t' = (120 - t)/20
        guard(t == 100)
        {
            t = 0;
            Machine.Heated();
        }
    }
    HeatUp90()
    {
        inv(t<= 90)
            t' = (120 - t)/20
        guard(t == 90)
        {
            t = 0;
            Machine.Heated();
        }
    }
}
```

شکل ۶: مدل اکتور Heater



شکل ۷: خودکاره‌ی ترکیبی مدل ماشین نوشیدنی

۴-۳- پل متحرک

در این مثال یک پل متحرک در نظر گرفته شده است که به طور پیش فرض در حالت بالا آمده قرار دارد. در صورت وجود ماشین در پشت پل، پل شروع به پایین آمدن می‌کند و زمانی که پل به طور کامل پایین آمده باشد، ماشین‌ها شروع به گذر می‌کنند. در صورت رد شدن تمام ماشین‌ها، پل دوباره شروع با بالا رفتن می‌کند و با آمدن ماشین‌های جدید این رویه دوباره تکرار می‌شود.

در این مدل دو اکتور تعریف CarDispatcher و DrawBridge تعریف شده است. CarDispatcher وظیفه‌ی مدل کردن ورود ماشین‌ها را دارد. این اکتور به صورت دوره‌ای یک ماشین به DrawBridge اضافه می‌کند. اکتور DrawBridge وظیفه‌ی کنترل پل را به عهده دارد. این اکتور دارای دو متغییر گسسته‌ی cars و bridgeStatus است. متغییر cars تعداد ماشین‌های پشت پل است و متغییر bridgeStatus حالت پل را مشخص می‌کند که در این مدل، پل دارای سه حالت بالاآمده، درحال تغییر و پایین آمده است که به ترتیب با مقادیر 0، 1 و 2 مشخص شده است. این اکتور همچنین دارای دو متغییر پیوسته‌ی degree و timer است. متغییر degree نشان دهنده‌ی زاویه‌ی دسته‌ی پل است که بین اعداد ۹۰ و ۰ متغییر است. متغییر timer یک متغییر کمکی برای مدل سازی خروج ماشین‌ها به صورت دوره‌ای است.

```
EnqueueCar()
{
    cars = cars +1;
    if(bridgeStatus == 0)
    {
        self.StartLowering();
    }
}

StartLowering()
{
    if(bridgeStatus == 0)
    {
        bridgeStatus = 1;
        inv(degree>=0)
            degree'= -10;
        guard(degree == 0)
        {
            bridgeStatus = 2;
            self.PassACar();
        }
    }
}
```

شکل ۸: متدهای EnqueueCar و StartLowering

در متد EnqueueCar به ماشین‌های پشت پل یک واحد اضافه می‌شود و در صورت بالا بودن پل، اکتور پیام StartLowering را به خودش ارسال می‌کند. در این پیام حالت پل به درحال تغییر، تغییر می‌کند. و یک رفتار فیزیکی برای مدل کردن پایین آمدن پل اجرا می‌شود که در پایان این رفتار، حالت پل به پایین آمده تغییر می‌کند و پیام PassACar به خود اکتور

ارسال می‌شود. متدهای EnqueueCar و StartLowering در شکل ۸ آورده شده است.

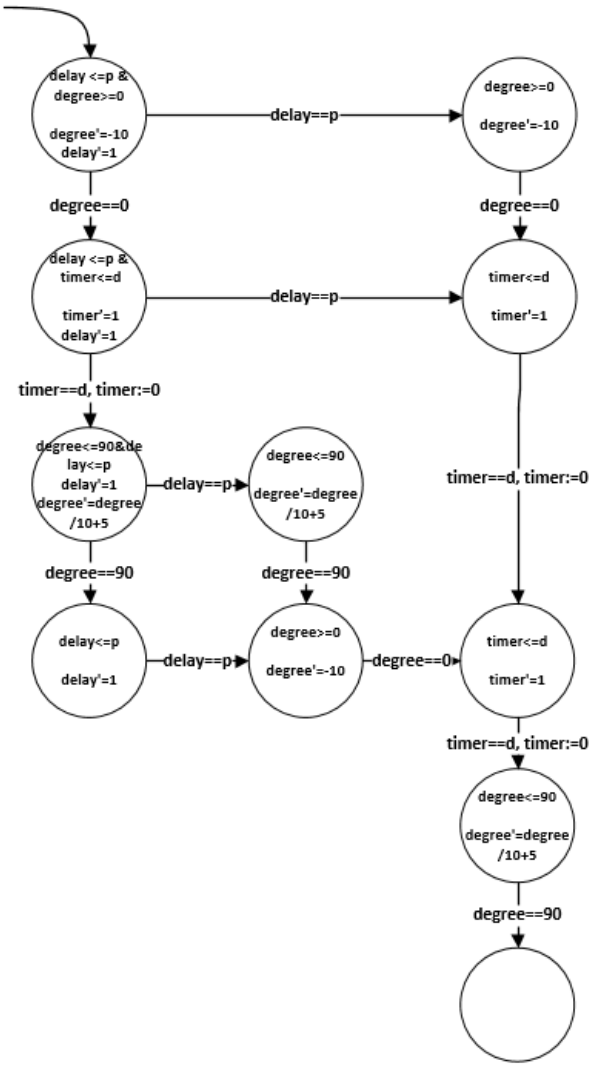
```
StartRaising()
{
    bridgeStatus = 1;
    inv(degree<=90)
        degree'= degree/100;
    guard(degree == 90)
    {
        bridgeStatus = 0;
        if(cars > 0)
            self.StartLowering();
    }
}

PassACar()
{
    inv(timer<=d)
        timer'=1;
    guard(timer == d)
    {
        timer = 0;
        cars = cars-1;
        if(cars <= 0)
            self.StartRaising();
        else
            self.PassACar();
    }
}
```

شکل ۹: متدهای PassACar و StartRaisng

متد PassACar به صورت دوره‌ای با استفاده از متغییر timer یک ماشین را از پل خارج می‌کند و پس از اتمام خروج در صورت وجود ماشین دیگری در صف، دوباره پیام PassACar را ارسال می‌کند و در صورت اتمام ماشین‌ها پیام StarRaising را ارسال می‌کند. این متد بالا آمدن پل را مدل می‌کند. پس از بالا آمدن پل، در صورت وجود ماشین، پل دوباره به شروع به پایین آمدن می‌کند. متدهای PassACar و StartRaising در شکل ۹ نشان داده شده است. خودکاره‌ی ترکیبی معادل برای دو ماشین در شکل ۱۰ آمده است. متغییر delay برای رفتار دوره‌ای CarDispatcher است.

۵-جمع‌بندی و کارهای آینده

مدل‌سازی مبتنی بر اکتور به دلیل ساختار پیمانه‌ای، اجازه‌ی تعریف مدل‌های پیچیده را از ترکیب مولفه‌های ساده‌تر، می‌دهد. در زبان HPalang اکتورهای تعریف شده به طور مستقل و هم‌روند اجرا شده و با استفاده از ارسال پیام‌های غیرسنکرون با یکدیگر ارتباط برقرار می‌کنند. همچنین هر اکتور می‌تواند چندین رفتار پیوسته برای مدل‌سازی رفتار فیزیکی تعریف کند. این امر باعث می‌شود به راحتی بتوان رفتارهای پیچیده‌ی سایبری را در کنار رفتارهای فیزیکی، مدل‌سازی کرد.



شکل ۱۰: خودکاره ترکیبی مدل DrawBridge

مدل معنایی تعریف شده برپایه خودکاره‌ی ترکیبی، تنها نشان دهنده‌ی رفتار فیزیکی سیستم در نتیجه‌ی رفتارهای سایبری است ولی خود رفتارهای سایبری در خودکاره‌ی نهایی وجود ندارند. یکی از کارهای آینده در این رابطه، ایجاد یک چارچوب تحلیل برای این زبان است، تا بتوان رفتارهای فیزیکی و سایبری را در کنار یکدیگر تحلیل کرد. همچنین با توجه به تعریف مدل معنایی بدون در نظر گرفتن رفتار فیزیکی واقعی سیستم، حجم مدل خودکاره‌ی ترکیبی با افزایش تعداد رفتارهای پیوسته، به طور نمایی افزایش پیدا می‌کند، ولی بیشتر این حالات با توجه به مشخصات رفتار فیزیکی سیستم، غیر قابل درسترس هستند. با تحلیل تقریبی رفتار فیزیکی مدل، می‌توان حجم مدل نهایی را به شدت کاهش داد.

در این نسخه‌ی زبان تنها بخش ارسال پیام غیرسکنرون برای مدل‌سازی رفتار شبکه تعریف شده است. یکی دیگر از کارهای آینده تعریف مفاهیمی چون تاخیر شبکه و مهلت پردازش پیام‌ها است تا

با توجه به نقش مهم شبکه در سیستم‌های سایبر-فیزیکی، مدل دقیق‌تری را بتوان تعریف کرد.

استفاده‌ی ترکیبی از متغییرهای پیوسته و گسسته نیز یکی دیگر از زمینه‌های تحقیق در این زبان است. در این نسخه امکان استفاده ترکیبی از متغییرهای پیوسته و گسسته در دستورات چون دستور شرطی و دستور رفتار پیوسته وجود ندارد.

مراجع

[1] A. H. Reynisson, M. Sirjani, L. Aceto, M. Cimini, A. Jafari, A. Ingólfsdóttir and S. H. Sigurdarson, "Modelling and simulation of asynchronous real-time systems using Timed Rebeca," *Science of Computer Programming*, vol. 89, pp. 41-68, 2014.

[2] R. Goebel, R. G. Sanfelice and A. R. Teel, "Hybrid dynamical systems," *IEEE Control Systems*, pp. 28-93, 2009.

[3] C. Ptolemaeus, System Design, Modeling, and Simulation: Using Ptolemy II, Ptolemy. org Berkeley, 2014.

[4] D. A. van Beek, W. Fokkink, D. Hendriks, A. Hofkamp, J. Markovski, J. M. van de Mortel-Fronczak and M. A. Reniers, "CIF 3: Model-Based Engineering of Supervisory Controllers," in *Tools and Algorithms for the Construction and Analysis of Systems - 20th International Conference, {TACAS} 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014. Proceedings*, 2014.

[5] W. Taha, "Acumen 2014 Reference Manual," Halmstad University, Department of Computer, 2014. [Online]. Available: http://bit.ly/Acumen-manual-2014. [Accessed 23 1 2017].

[6] G. Frehse, C. L. Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang and O. Maler, "SpaceEx: Scalable Verification of Hybrid Systems," in *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*, 2011.