

یک زبان مدل سازی مبتنی بر اکتور برای سیستم های سایر-فیزیکی

ایمان جهاننیده

مبتنی بر مدل، یکی از روش های اصلی در طراحی چنین سیستم هایی است. این مدل ها علاوه بر تعیین مشخصات سیستم، توانایی شبیه سازی و بررسی آن را فراهم می کنند.

یکی از پایه ای ترین زبان های مدل سازی برای سیستم های سایر-فیزیکی، خودکارهای ترکیبی^۱ است. این مدل می تواند رفتارهای پیوسته (فیزیکی) و رفتارهای گسسته (سایبری) را توصیف کند. به طور خلاصه یک خودکارهای ترکیبی دارای چندین مکان برای مدل کردن حالات گسسته سیستم و دارای چندین متغیر پیوسته و معادلات زمانی آن ها برای بیان رفتار پیوسته سیستم است. با این همه مدل سازی رفتارهای پیچیده سایبری و مدل سازی رفتارهای شبکه در این زبان سخت است.

در مقابل Timed Rebeca [1] یک زبان برای مدل سازی سیستم های غیرسنکرون بی درنگ 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 مدل سازی رفتار پیوسته، مفهومی ارائه نمی دهد.

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

- مدل معنایی صوری مبتنی بر خودکارهای ترکیبی
- مدل سازی ارسال و دریافت پیام به صورت غیرسنکرون
- مدل سازی زمان پردازشی بخش سایر
- مدل سازی 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^۱

چکیده- یکی از چالش های مدل سازی سیستم های سایر-فیزیکی^۱، ادغام رفتارهای فیزیکی با رفتارهای محاسباتی و شبکه است و با توجه به این که سیستم های سایر-فیزیکی معمولاً دارای مولفه های توزیع شده هستند، مدل های مبتنی بر اکتور، چارچوب مناسبی را برای مدل سازی چنین سیستم هایی فراهم می کنند. در این پروژه یک زبان مدل سازی صوری مبتنی بر اکتور برای سیستم های سایر-فیزیکی ارائه شده است، که توانایی توصیف رفتارهای فیزیکی، محاسباتی و شبکه را فراهم می کند.

کلمات کلیدی- سیستم های سایر-فیزیکی، مدل سازی، مدل سازی مبتنی بر اکتور

۱- مقدمه

سیستم های سایر-فیزیکی از ادغام فرایندهای محاسباتی و فیزیکی ایجاد می شوند. در این سیستم ها، رایانه های نهفته^۲ از طریق شبکه، فرایندهای فیزیکی را کنترل می کنند که معمولاً همراه با حلقه های بازخورد^۳ است، به این معنی که فرایندهای فیزیکی بر فرایندهای محاسباتی تاثیر می گذارند و بالعکس. به همین خاطر برای تحلیل و توسعه این سیستم ها نیاز به آگاهی از تعامل و اشتراک بین سیستم رایانه ای، نرم افزار، شبکه و فرایندهای فیزیکی است.

سیستم های سایر-فیزیکی چند تفاوت کلیدی با سیستم های نرم-افزاری دارند. در سیستم های نرم افزاری زمان اجرای یک دستور تنها مرتبط با کارایی سیستم است و نه درستی آن ولی در مقابل در سیستم های سایر-فیزیکی زمان اجرای یک دستور می تواند برای درستی سیستم حیاتی باشد. همچنین فرایندهای فیزیکی ترکیب رخداد چندین رویداد همزمان هستند، بر خلاف فرایندهای نرم افزاری که به صورت گام های متوالی اجرا می شوند.

با توجه به کاربرد سیستم های سایر-فیزیکی در زمینه های ایمنی-مهم^۴، ارزیابی و بررسی این سیستم ها به شدت حیاتی است. روش های

^۱ Safety-Critical

^۲ Hybrid Automata

^۳ Transition System

^۱ Cyber-Physical

^۲ Embedded

^۳ Feedback Loop

یک سیستم گذار یک چندتایی به صورت $(S, \rightarrow, I, L, s_0)$ 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.

۲- مسئله و کارهای مشابه

۱-۲- خودکاری ترکیبی

یک خودکاری 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 متغیرهای پیوسته است.
- l_0 مکان شروع و v_0 مقادیر اولیه متغیرهای 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

متغیرهای V پس از پرش است.

- $I : Loc \rightarrow B(V)$ مقادیر مجاز متغیرها را در هر مکان

مشخص می‌کند.

- $F : Loc \rightarrow B(V \cup \dot{V})$ محدودیت‌ها روی متغیرها و

مشق آن‌ها توصیف می‌کند و رفتار پیوسته در هر مکان را

مشخص می‌کند.

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.

premise

conclusion

شکل ۱: ساختار قانون SOS

زبان‌های مختلفی برای مدل‌سازی و تحلیل و درستی‌یابی سیستم‌های سایبر-فیزیکی ارائه شده است. این زبان‌ها در صورتی 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-سکرو^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 صورتی برای Ptolemy تعریف نشده است.

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

زبان Acumen [4] یک زبان مدل‌سازی و شبیه‌سازی برای سیستم‌های 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

^{۱۰} Data Flow

^{۱۱} Director

^۷ Finite Transition System

^۸ Synchronous-Reactive

^۹ Discrete Event

تعیین مدل‌های مختلف سیستم استفاده کرد. رفتارهای فیزیکی به صورت مشتق متغیرها 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 سایر-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 سایر-فیزیکی است. همچنین با توجه به ساختار مبتنی بر اکتور، این زبان یک روش طراحی پیمانه‌ای را ارائه می‌دهد. این زبان اجازه تعریف رفتارهای سایر-فیزیکی به صورت دستوری فراهم می‌کند که یکی از روش‌های معمول برای بیان رفتار-های سیستمی است.

هرچند روش‌های درستی‌یابی در زمینه‌ی سیستم‌های سایر-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، اجازه‌ی درستی‌یابی مدل‌های این زبان را ارائه می‌دهد.

۳- تعریف زبان

۱-۳- نشانه‌گذاری‌ها

در تعریف زبان HPalang دنباله خالی با نماد ϵ نمایش داده شده است و $|h|T$ 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 $f: X \rightarrow Y$ از نماد $f[x \rightarrow y]$ به معنی تابع $\{(x, y) \mid (x, y) \in f \mid a \neq x\} \cup \{(a, b) \in f \mid a \neq x\}$ استفاده شده است.

۲-۳- نحو^{۱۴}

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

۱-۲-۳- اکتور

یک اکتور به صورت یک چندتایی به فرم $(id, vars, mthds)$ تعریف می‌شود که id شناسه، $vars$ مجموعه‌ای متغیرهای گسسته و پیوسته و $mthds$ 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.

۲-۲-۳- دستورات

در این قسمت دستورات این زبان تعریف شده‌اند. $DVar$ و $CVar$ به ترتیب مجموعه‌ی متغیرهای گسسته و پیوسته هستند و $DExpr$ و $CExpr$ به ترتیب عبارات ریاضی شامل متغیرهای گسسته و عبارات ریاضی شامل متغیرهای 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 متغیر گسسته است. از نماد $var = expr$ برای نمایش استفاده شده است.

$CAssignment = CVar \times CExpr$ عملیات انتصاب یک مقدار پیوسته به یک متغیر 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 $if\ expr\ then\ \sigma\ else\ \sigma'$ 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.

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

است. $actions$ دنباله‌ی دستوراتی است که پس از انجام گذار اجرا می- شوند. 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.

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 سیستم گذار 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.

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

$delayVar : ID \rightarrow CVar$ که $delayVar(x)$ متغیر 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_l و q_h 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 سراسری یک مدل به صورت یک تابع $l : ID \rightarrow DS \times CB$ نمایش داده می‌شود که $DS = (Var \rightarrow Val) \times Queue \times Stat^*$ و $CB = CExpr \times ODE^* \times CExpr \times Queue \times Stat^*$. 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 متغیرهای 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

$$\frac{l(x) = (ds, cs) \wedge ds = (v, \varepsilon, m | T, \varepsilon) \wedge \neg v(suspended) \wedge l'(x) = (ds', cs') \wedge ds' = (v', q'_h, q'_l, \varepsilon) \wedge \forall i \in ID \setminus \{x\} (l(i) = (v'', q''_h, q''_l, \varepsilon, cs) \wedge l'(i) = (v'', q''_h, q''_l, \varepsilon, cs^*)) \wedge l[x[ds] \mapsto (v, \varepsilon, T, body(m))] \Rightarrow^* l'}{l \xrightarrow{T} l'}$$

High Priority Message Take

$$\frac{l(x) = (ds, cs) \wedge ds = (v, m | T, q_l, \varepsilon) \wedge l'(x) = (ds', cs') \wedge ds' = (v', q'_h, q'_l, \varepsilon) \wedge \forall i \in ID \setminus \{x\} (l(i) = (v'', q''_h, q''_l, \varepsilon, cs) \wedge l'(i) = (v'', q''_h, q''_l, \varepsilon, cs^*)) \wedge l[x[ds] \mapsto (v, T, q_l, body(m))] \Rightarrow^* l'}{l \xrightarrow{T} l'}$$

```
<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> ')' '{' <stat-list> '}'
    'guard' '(' (<cexpr> ')' '{' <stat-list> '}'
<code-list> ::= (var ' ')*
<main> ::= 'main' '{' (<send> ';')* '}'
<message> ::= <identifier>
<actor-id> ::= <identifier>
<var> ::= <identifier>
<type> ::= 'int' | 'real'
```

شکل ۲: گرامر زبان HPalang به فرم EBNF

۳-۲-۳- 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 متغیرها 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.

Message Send

$$\begin{aligned} l(x) &= (ds_x, cs_x) \\ l(y) &= (ds_y, cs_y) \\ ds_x &= (v^x, q_h^x, q_l^x, (y, m)|\sigma^x) \\ ds_y &= (v^y, q_h^y, q_l^y, \sigma^y) \end{aligned}$$

$$\begin{aligned} \frac{}{l \xrightarrow{\tau} l[x \mapsto (ds'_x cs_x)][y \mapsto (ds'_y, cs_y)]} \\ ds'_x &= (v^x, q_h^x, q_l^x, \sigma^x) \\ ds'_y &= (v^y, q_h^y, q_l^y \oplus m, \sigma^y) \end{aligned}$$

۳-۳-۴ - سیستم گذار میانی

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

- l مجموعه حالات سراسری است (مجموعه ای تمام توابع از شناسه ای اکتورها به حالات محلی)
- \rightarrow کوچکترین رابطه ای است که توسط قوانین SOS بالا تعریف شده است
- s_0 حالات اولیه است که متغیرهای گسسته مقدارهای اولیه شده اند و پیام های درون بلوک main در صف اولیت پایین اکتورهای مشخص شده گذاشته شده است.
- L برچسب هایی است که مطابق قوانین بالا مشخص شده است.

۳-۳-۵ - خودکاری ترکیبی

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

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

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

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

Continuous Behavior Expiration

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

Continuous Variable Assignment

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

Discrete Variable Assignment

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

Conitional True

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

Conitional False

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

Resume Statement

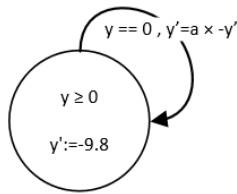
$$\begin{aligned} \frac{l(x) = (ds, cs) \wedge ds = (v, q_h, q_l, resume|\sigma)}{l \xrightarrow{\tau} l[x \mapsto (ds', cs)]} \\ ds' &= (v[suspended := false], q_h, q_l, \sigma \oplus v(comd)) \end{aligned}$$

Continuous Behavior Statement

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

Delay Statement

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



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 سانتی‌گراد و ۳۰۰ سی‌سی و دما و حجم نوشیدنی قهوه به ترتیب ۹۰ درجه‌ی سانتی‌گراد و ۲۰۰ سی‌سی است.

```
actor Machine
{
  int orderType;

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

  Heated()
  {
    if(orderType == 0)
      Filler.Fill1300();
    else
      Filler.Fill1200();
  }
  Filled()
  {
    User.ReceiveOrder();
  }
}
```

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 متغیر 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

- مجموعه‌ی متغیرهای پیوسته‌ی مدل V

- $loc_0 = s_0$

- مقداردهی اولیه به متغیرهای پیوسته v_0

- 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 برخورد 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 متغیر 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 ضریب 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 خودکارهی ترکیبی در شکل ۴ آمده است.

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 می کند و زمانی که پل به طور کامل پایین آمده باشد، ماشین ها شروع به گذر می کنند. در صورت رد شدن تمام ماشین ها، پل دوباره شروع به بالا رفتن می کند و با آمدن ماشین های جدید این رویه دوباره تکرار می شود.

در این مدل دو اکتور تعریف `CarDispatcher` و `DrawBridge` تعریف شده است. `CarDispatcher` وظیفه ی مدل کردن ورود ماشین ها را دارد. این اکتور به صورت دوره ای یک ماشین به `DrawBridge` اضافه می کند. اکتور `DrawBridge` وظیفه ی کنترل پل را به عهده دارد. این اکتور دارای دو متغیر گسسته ی `cars` و `bridgeStatus` است. متغیر `cars` تعداد ماشین های پشت پل است و متغیر `bridgeStatus` حالت پل را مشخص می کند که در این مدل، پل دارای سه حالت بالا آمده، در حال تغییر و پایین آمده است که به ترتیب با مقادیر 0، 1 و 2 مشخص شده است. این اکتور همچنین دارای دو متغیر پیوسته ی `degree` و `timer` است. متغیر `degree` نشان دهنده ی زاویه ی دسته ی پل است که بین اعداد ۹۰ و ۰ متغیر است. متغیر `timer` یک متغیر کمکی برای مدل سازی خروج ماشین ها به صورت دوره ای 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: مندهای `EnqueueCar` و `StartLowering`

در متد `EnqueueCar` به ماشین های پشت پل یک واحد اضافه می شود و در صورت بالا بودن پل، اکتور پیام `StartLowering` را به خودش ارسال می کند. در این پیام حالت پل به در حال تغییر، تغییر می کند. و یک رفتار فیزیکی برای مدل کردن پایین آمدن پل اجرا می شود که در پایان این رفتار، حالت پل به پایین آمده تغییر می کند و پیام `PassACar` به خود اکتور

مطلوب، متغیر `t` صفر می شود و پیام `Heated` به `Machine` فرستاده می شود. در متد `Heated`، `Machine` با توجه به نوع نوشیدنی درخواستی، پیام `Fill200` یا `Fill300` را به `Filler` ارسال می کند. `Filler` با دریافت یکی از این دو پیام، یک رفتار پیوسته برای مدل کردن پر شدن لیوان تعریف می کند. در این رفتار پس از رسیدن به حجم مناسب، پیام `Filled` به `Machine` ارسال می شود. در متد `Filled`، `Machine` پیام `RecieveOrder` را به `User` می فرستاد و `User` در این متد دوباره یک `Order` جدید می دهد. دلیل استفاده از دو متد در اکتورهای `Filler` و `Heater`، عدم تعریف شدن پارامتر ورودی برای متدها، در این نسخه است. در شکل ۵ و ۶ مدل اکتور `Machine` و `Heater` آمده است. همچنین در شکل ۷ خودکاره ی ترکیبی معادل مدل نشان داده شده است.

```

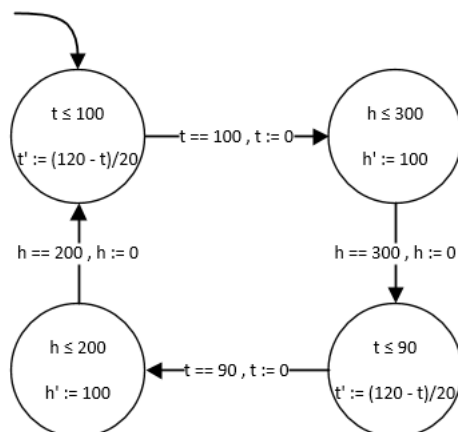
actor Heater
{
    real t;

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

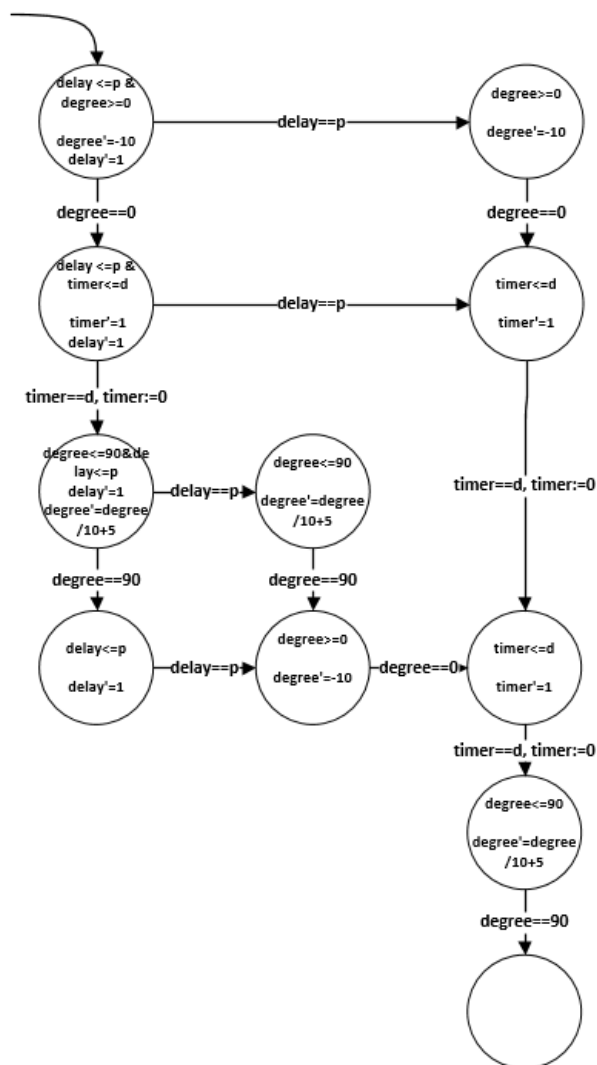
    HeatUp90()
    {
        inv (t <= 90)
        t' = (120 - t) / 20
        guard (t == 90)
        {
            t = 0;
            Machine.Heated();
        }
    }
}

```

شکل ۶: مدل اکتور `Heater`



شکل ۷: خودکاره ی ترکیبی مدل ماشین نوشیدنی



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 غیرسکرون برای مدل‌سازی رفتار شبکه تعریف شده است. یکی دیگر از کارهای آینده تعریف مفاهیمی چون تاخیر شبکه و مهلت پردازش پیام‌ها است تا

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 StartRaising

متد PassACar به صورت دوره‌ای با استفاده از متغیر 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 StartRaising 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. متغیر 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 HPalung اکتورهای تعریف شده به طور مستقل و هم‌روند اجرا شده و با استفاده از ارسال پیام‌های غیرسکرون 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.

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

استفاده‌ی ترکیبی از متغیرهای پیوسته و گسسته نیز یکی دیگر از زمینه‌های تحقیق در این زبان است. در این نسخه امکان استفاده ترکیبی از متغیرهای 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. Ingólfssdóttir 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.