یک روش مبتنی بر دامنه برای شبیه‌سازی ناظر در کنترل نظارتی سیستم‌های گسسته

سید مرتضی بابامیر
دانشگاه کاشان، گروه مهندسی کامپیوتر
babamir@kashanu.ac.ir

چکیده: در کنترل نظارتی سیستم‌های گسسته، یک پایشگر پاسخ‌های سیستم به وقایع محیطی را پایش می‌کند تا اگر پاسخ سیستم نامطلوب باشد، یک موقعیت ناامن یا بحرانی را به کاربر گزارش کند. منظور از پاسخ نامطلوب، پاسخی از سیستم است که باعث نقض نیاز کاربر سیستم شود. تاکنون چندین روش برای مدل‌سازی و شبیه‌سازی کنترل سرپرستی سیستم‌های گسسته ارائه شده‌اند، اما فقدان یک روش سیستماتیک که متکی به داده‌های دامنه مسأله باشد، وجود دارد. به منظور ارائه یک روش مبتنی بر دامنه، ما از یک روش سه مرحله‌ای استفاده می کنیم. در مرحله اوّل، با داده‌های گسسته شروع می‌کنیم. داده‌های گسسته، عناصر اوّلیه محیط سیستم‌های گسسته است و به وسیله کاربران سیستم به عنوان یک وسیله اندازه‌گیری نیازهایشان استفاده می‌شود. نیازها، در سیستم‌های حساس به ایمنی، مانند سیستم‌های پزشکی و هوافضا نقش حیاتی ایفا می‌کند. ما از داده‌های گسسته برای تعریف رخدادها و شروطی استفاده می‌کنیم که در تعریف نیازها استفاده می‌شود. پس از استخراج رخدادها و شروط از روی داده‌های دامنه مسأله و تعریف نیازها، یک اتوماتای پتری ساخته می‌شود. این اتوماتا برای تعیین نقض نیازهای کاربران سیستم در مرحله دوّم استفاده می‌شود. اتوماتای پتری هسته پایشگر را تشکیل می‌دهد و برای تشخیص پاسخ‌های نامطلوب سیستم به نیازهای کاربران استفاده می‌شود. در قدم سوّم، شبیه‌سازی هنگام اجرای مشاهده‌گر ارائه می‌شود که در آن از فناوری‌های چندخطی و چندنخی و چند وظیفه‌ای کتابخانه TPL از ماکروسافت استفاده می‌شود. در خاتمه، سیستم حفاظت قطار به عنوان یک مورد مطالعه از سیستم‌های گسسته همروند مطرح می‌شود تا نحوه به‌کارگیری مراحل روش پیشنهادی برای مدل‌سازی و شبیه‌سازی مشاهده‌گر نشان داده شود. نتایج شبیه‌سازی بر اساس پیاده‌سازی روی رایانه چند هسته تحلیل می‌شود.

واژه‌های کلیدی: کنترل نظارتی، شبیه‌سازی، مدل‌سازی، توصیف مبتی بر دامنه

۱- مقدمه

یک سیستم گسسته، سیستمی با تعداد شمارا از حالات است. این بدین معنی است که رفتار یک سیستم گسسته می‌تواند به‌طور منظم از طریق حالاتش به صورت انتزاعی بیان شود. در نتیجه، یک سیستم گسسته اغلب با یک ماشین حالت، مدل و تحلیل می‌شود. یک سیستم را سیستم رخداد گسسته[1] گویند اگر حالات آن بر طبق رویداد رخدادهای گسسته تغییر کند و گذار حالت در زمان‌های گسسته در پاسخ به رخدادها انجام شود. از نمونه سیستم‌های رخداد گسسته می‌توان شبکه‌های رایانه‌ای، سیستم‌های ارتباطی، سیستم‌های ترافیک شهری و سیستم‌های گردش کار را نام برد.

از آنجا که پاسخ یک سیستم رخداد گسسته ممکن است نیازهای کاربرانش را نقض کند، یک کنترل‌کننده به نام سرپرست[2] برای سرپرستی رفتار سیستم لازم است. به این منظور، نظریه کنترل سرپرستی[3] به عنوان یک روش رسمی برای ساخت کنترل‌کننده سیستم‌های رخداد گسسته تدوین شد [۱]. کنترل‌کننده یا سرپرست، مسؤولیت پایش رفتار سیستم و هدایت آن را به یک حالت امن در زمانی که نیاز

[1] تاریخ ارسال مقاله : ۱۳۸۹/۹/۲۷
تاریخ پذیرش مقاله : ۱۳۹۱/۸/۳۰
نام نویسنده مسؤول : سید مرتضی بابامیر
نشانی نویسنده مسؤول: ایران – کاشان – دانشگاه کاشان – دانشکده مهندسی – گروه مهندسی کامپیوتر

کاربران نقض می شود، به عهده دارد.

از طرف دیگر، برای مطالعه یک سیستم با استفاده از شبیه‌سازی، ما نخست یک مدل انتزاعی از سیستم که در برگیرنده ویژگی‌های سیستم است، ارائه می دهیم. سپس این مدل را با نوشتن برنامه‌هایی که اجرای آنها رفتار مدل را شبیه‌سازی می‌کند، پیاده‌سازی می‌کنیم. شبیه‌سازی یک سیستم رخداد گسسته، مدل سیستمی را مشخص می‌کند که در آن تغییرات در زمان‌های گسسته اتفاق می‌افتد. برای مثال، در یک مدل از یک شبکه رایانه‌ای، ورود یک پیام به شبکه، یک تغییر در حالت مدل است. از آنجا که چنین تغییراتی مهم هستند، لازم است رفتار مدل را در زمانی که یک تغییر رخ می‌دهد، مشاهده کنیم.

این مقاله تصمیم دارد تا به مدل‌سازی بخش پایشگر کنترل سرپرستی با استفاده از داده‌های دامنه مسأله که به وسیله کاربر سیستم بیان می‌شود، بپردازد. شبیه‌سازی چنین پایشگری با مسأله مشاهده و وارسی رفتار سطح پایین هنگام اجرای سیستم در مقابل توصیف‌های سطح بالای نیازهای کاربران مواجه است، زیرا این دو به علت متفاوت بودن ماهیتشان، قابل مقایسه نیستند. به این منظور، رخدادها و نیازهای کاربران را باید از داده‌های استفاده شده در دامنه مسأله منتزع کرد و سپس به فعالیت های هنگام اجرای سیستم نگاشت نمود. اگرچه روش‌های متعددی تاکنون برای مدل‌سازی و شبیه‌سازی کنترل سرپرستی سیستم‌های گسسته ارائه شده است، فقدان یک روش سیستماتیک که متکی به داده‌های دامنه مسأله باشد، به چشم می‌خورد.

پس از توصیف رسمی رخدادها و نیازها، یک اتوماتا که معرف نقض نیازهاست، ساخته می‌شود تا یک توصیف مبتنی بر حالت از نقض نیازها ارائه شود. این توصیف به عنوان پلی بین توصیف‌های مبتنی بر رخداد و مراحل شبیه‌سازی در نظر گرفته می‌شود. در حقیقت، این اتوماتا به عنوان یک مدل انتزاعی برای پیاده‌سازی مرحله شبیه‌سازی استفاده می‌شود. این اتوماتا که شامل هسته پایشگر است برای تعیین حالاتی که سیستم نباید وارد آنها شود، استفاده می‌شود.

دو نوع همروندی وجود دارد: (۱) همروندی در اجرا که به وسیله پاسخ سیستم، رخدادهای محیط سیستم و کنترل سرپرستی ایجاد می‌شود و (۲) همروندی بین رخدادها. تغییر سرعت و تغییر وضعیت ترمز قطار در سیستم حفاظت قطار، نمونه‌ای از همروندی در رخدادهاست. این سیستم هدایت قطارها در نواحی "هشدار" ، "معمولی" و "آزاد" را از طریق توصیه دستورالعمل‌هایی برای تنظیم سرعت و ترمز به عهده می‌گیرد. بنابراین، مکانیسمی برای اداره همروندی در مرحله شبیه‌سازی باید استفاده شود. در این راستا، ما شبیه‌سازی پایشگر را با استفاده از کتابخانه TPL ماکروسافت روی رایانه چند هسته با زبان C# [۲و۳] انجام می‌دهیم. ما از نخ ها و توابع TPL برای اداره همروندی در شبیه‌سازی رفتار سیستم رخداد گسسته استفاده می‌کنیم و سپس کارایی آنها را در یک مورد مطالعه مقایسه می‌کنیم. در خاتمه، روشمان را برای مدل‌سازی و شبیه‌سازی بخش پایشگر در کنترل سرپرستی سیستم حفاظت قطار به کار می‌بریم. کنترل سرپرستی، وارسی ارضا نیازهای کاربران به وسیله سیستم را به عهده دارد.

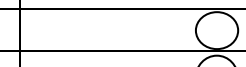
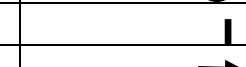
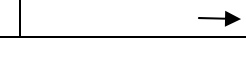
این مقاله به صورت زیر ادامه می‌یابد: در بخش دوّم ابتدا تعاریف رخدادها و ثابت‌ها[۴] را به صورت رسمی بیان و سپس بر اساس این تعاریف، نیازهای ایمنی[۵] را تعریف می‌کنیم. در بخش سوّم، از تعاریف رخدادها و ثابت‌ها برای توصیف مبتنی بر حالت سیستم استفاده می‌کنیم. در بخش چهارم، به طراحی شبیه‌ساز با در نظر گرفتن توصیف مبتنی بر حالت، می‌پردازیم. برای نشان دادن این که چگونه باید روش ما برای یک مسأله خاص به کار رود، یک سیستم حساس به ایمنی را به عنوان یک مورد مطالعه در بخش پنجم مطرح می‌کنیم. در بخش ششم، به ارزیابی کارآیی روشمان می پردازیم. در بخش هفتم،کارهای مرتبط را مطالعه می‌کنیم تا تفاوت‌ها و تشابهات آنها را با روش پیشنهادی نشان دهیم. در بخش هشتم، نتایجی، که از روش پیشنهادی به دست می‌آید، مشخص می‌کنیم.

۲- توصیف رخدادها و ثابت‌ها

از داده‌های دامنه مسأله که بر حسب واژگان کاربر سیستم بیان می‌شود، یک محیط را با ویژگی‌هایش توصیف می‌کنیم که در آن یک ویژگی، مشخصه مهّمی از محیط سیستم است. پس از آن، رخدادها و ثابت‌ها را با استفاده از این ویژگی‌ها تعریف می‌کنیم. در نهایت، نقض نیازهای کاربران سیستم را با یک رخداد و تعدادی ثابت توصیف می‌کنیم. یک رخداد معرّف تغییری جدی در مقدار یک

ویژگی است، امّا یک ثابت معرف عدم تغییر در یک ویژگی دیگر در زمان یک رخداد است. برای مثال، در سیستم کنترل قطار، قطار محیط سیستم است و "سرعت" و "ناحیه" دو ویژگی از این محیط هستند. جمله "سرعت قطار نباید در هنگام ورود به ناحیه هشدار بالا باشد"، یک نیاز ایمنی روی این دو ویژگی است که نباید نقض شود. عبارت " ورود قطار به ناحیه هشدار" یک رخداد و عبارت "سرعت بالا" یک ثابت است. به عبارت دیگر، این رخداد نیاز کاربر را نقض می کند، در صورتی که ثابت "سرعت بالا" برقرار باشد. در ادامه، به تعریف رسمی رخدادها، ثابت ها و نیازهای کاربر با استفاده از ویژگی‌ها می‌پردازیم. برای خوانایی بیشتر، جدولی برای توصیف نمادهایی که در بخش‌های بعدی این مقاله استفاده می‌شود، ارائه می‌شود (جدول ۱).

جدول (۱): شرح نمادهای مورد استفاده

نماد	توصیف
$\mathbb{R}$	مجموعه اعداد حقیقی
$\mathbb{R}_+$	مجموعه اعداد حقیقی مثبت
$\leq$	رابطه ترتیب
Pr	ویژگی
$a \leq a$	رابطه انعکاس
$a \leq b \ \& \ b \leq a \Rightarrow a=b$	رابطه پادمتقارن
$a \leq b \ \& \ b \leq c \Rightarrow a \leq c$	رابطه تعدی
$\varnothing$	مجموعه تهی
$I_{i,p}$	iامین بازه ویژگی p
$(In_{i,p})_\tau$	یک شاخص روی بازه i از ویژگی p در زمان τ
$@T(In_{i,p})_\tau$	یک رخداد. iامین بازه ویژگی p در زمان τ برقرار می‌شود
$@F(In_{i,p})_\tau$	یک رخداد. از iامین بازه ویژگی p در زمان τ خارج می‌شویم
$t(In_{j,q})_{\tau,\tau+1}$	ثابت برقرار. iامین بازه ویژگی q در زمان‌های $\tau,\tau+1$ برقرار است و تغییر نمی‌کند
$f(In_{j,q})_{\tau,\tau+1}$	ثابت برقرار. iامین بازه ویژگی q در زمان‌های $\tau,\tau+1$ برقرار نیست و تغییر نمی‌کند
$\Sigma_{i,r} t(In_{i,r})_{\tau,\tau+1} / \Sigma_{i,r} f(In_{i,r})_{\tau,\tau+1}$	ثابت‌های برقرار / ثابت‌های نابرقرار
$\Sigma_{i\geq 1}@T(In_{i,p})_\tau$ or $\Sigma_{i\geq 1}@F(In_{i,p})_\tau$	رخدادهای همروند
$\{P_1, P_2, \ldots, P_m\}$	مجموعه متناهی از مکان‌ها در شبکه پتری
$\{T_1, T_2, \ldots, T_n\}$	مجموعه متناهی از گذارها در شبکه پتری
$\{F_1, F_2, \ldots, F_k\}$	مجموعه متناهی از قوس‌های جهت‌دار از مکان‌ها به گذارها و بالعکس
$M: P \rightarrow N$	رابطه نشانه‌گذاری که معرف تعداد نشانه‌ها (N) در یک مکان (P) است
$m_0(p)$	نشانه‌گذاری اولیه مکان p
$m_j(p)$	jامین نشانه‌گذاری مکان p
$\bullet T_i$	مکان ورودی به گذار T_i
$T_i\bullet$	مکان خروجی از گذار T_i
TPL	کتابخانه TPL ماکروسافت
API	واسط برنامه کاربردی
SRV	نقض نیازهای ایمنی (Safety Requirement Violation)
	مکان تهی در شبکه پتری
	مکان دارای نشانه در شبکه پتری
	گذار در شبکه پتری
	قوس در شبکه پتری

۲-۱- مدل سازی داده‌ها

محیط سیستم به وسیله تعدادی ویژگی مشخص می‌شود که در آن هر ویژگی که با Pr تعیین می شود. یک مجموعه خوش‌ترتیب از مقادیر منطقی، شمارشی، صحیح یا اعشاری است. خصیصه ترتیب، مجموعه منطقی {درست، نادرست} را از مجموعه منطقی {نادرست، درست) متمایز می‌کند. بنابراین، در این مجموعه‌ها، یک خصیصه خوش ترتیب با ترتیب نوشتن عناصر آنها مشخص می شود. یک مجموعه شمارشی شامل مقادیری است که با اسامی، مانند مجموعه ناحیه = {هشدار، معمولی، آزاد} مشخص می‌شود. یک مجموعه خوش‌ترتیب، یک مجموعه کاملاً (به طور خطی) مرتب است اگر بتوان اوّلین عنصر آن مجموعه و هر زیر مجموعه‌اش را مشخص کرد [۴]. برای مثال، مجموعه اعداد صحیح $\{\ldots\text{-}2,\text{-}1,0,1,2,\ldots\}=\mathbb{Z}$ که یک مجموعه مرتب خطی است، خوش‌ترتیب نیست، زیرا اوّلین عنصر آن را نمی‌توان مشخص کرد، اما مجموعه اعداد مثبت $\{1,2,3,\ldots\}=\mathbb{Z}^{+}$ یک مجموعه خوش‌ترتیب است، زیرا می‌توان اوّلین عنصر این مجموعه و هر زیر مجموعه آن را تعیین کرد. رابطه ترتیب "<" یک رابطه روی مجموعه Pr است که سه خاصیت P_1(انعکاس)، P_2 (پادمتقارن) و P_3 (تعدی) را ارضا کند.

P1. $a \in Pr, a \le a$
P2. $a, b \in Pr$, if $(a \le b$ and $b \le a) \Rightarrow a=b$
P3. $\forall a, b, c \in Pr$, if $(a \le b$ and $b \le c) \Rightarrow a \le c$.

تعریف: فرض کنید a و b دو عدد صحیح یا حقیقی متمایز باشند که a<b . یک بازه با نقاط نهایی a و b به عنوان یک بازه "بسته- باز" از a به b تعریف می‌شود و با [a ...b) نمایش داده می شود. بر اساس دامنه مسأله، هر Pr به بازه های منفصل با رابطه "<" شکسته می شود؛ به طوری که در دو بازه متوالی مانند [a…b) و [b…c) سوپرمم اوّلین بازه، می نیمم دومین بازه باشد. برای مثال، فرض‌کنید که سرعت یک ویژگی محیطی باشد که مقدارش به حداکثر ۱۰ مایل در ساعت می‌رسد و مقادیر ۳ ، ۷/۵ و ۱۰ مقادیر مرزی هستند. بر اساس این مقادیر دامنه، سرعت را به بازه‌های [0…3) و [3…7.5) و [7.5…10] تقسیم می‌کنیم. وقتی یک مقدار ورودی از محیط سیستم دریافت می‌شود، با می‌نیمم مقدار هر بازه مقایسه می‌شود تا بازه متناظر مشخص شود. این بازه، بازه‌ای است که مقدار خوانده‌شده از ورودی درآن قرار دارد. اکنون به تعریف ویژگی‌های بازه‌ها روی مقادیر صحیح یا اعشاری می‌پردازیم:

$$Pr_p = \{a_0, a_1, \ldots, a_n\} \equiv \bigcup_{i=1}^{k} I_{i,p} \qquad (۱)$$

که Pr_p معرف ویژگی p و یک مجموعه خوش‌ترتیب است و $I_{i,p}$ معرف i امین بازه بسته باز از این ویژگی است.

$$I_{1,p} = [a_0, \ldots, a_j), \ldots, I_{k,p} = [a_m, \ldots, a_n) \qquad (۲)$$

که $I_{i,p}$ یک عدد اعشاری یا یک عدد صحیح است و i متعلق به مجموعه [1…k] و p یک ویژگی است. رابطه (۲) معرف این است که هر $I_{i,p}$ یک بازه روی Pr_p است که می‌نیمم و سوپرمم مقدارش بر اساس دامنه مسأله و به وسیله کاربر سیستم تعریف می‌شود. برای مثال، برای یک وسیله نقلیه که با سرعت حداکثر ۱۰ مایل در ساعت حرکت می‌کند، بازه های[1…5)، [5…7) و [7…10] تعریف می‌شوند که به ترتیب معرف سرعت‌های "کم" ، "متوسط" و "زیاد" است.

یک بازه، یک مجموعه شامل دو خصوصیت است: (۱) یک مجموعه غیرتهی است که به وسیله رابطه (۳) به صورت رسمی تعریف می‌شود و (۲) هیچ عنصر مشترکی با دیگر مجموعه‌ها ندارد که به وسیله رابطه (۴) به صورت رسمی تعریف می‌شود. خصوصیت خوش‌ترتیبی یک ویژگی با رابطه (۵) بیان می‌شود و معرف این است که هر زیر مجموعه از یک ویژگی، خوش‌ترتیب است.

$$\forall i:\ I_{i,p} \neq \varnothing \qquad (۳)$$

$$\forall i, j:\ I_{i,p} \cap I_{j,p} = \varnothing : i \neq j,\ I_{i,p}, I_{j,p} \in Pr_p \qquad (۴)$$

$$a < b \Rightarrow a \in I_{i,p}, b \in I_{j,p} \text{ where } i < j \text{ or } a,b \in I_{i,p} \qquad (۵)$$

پس از آن که بازه‌ها تعیین شدند، می‌توانیم "شاخص" را تعریف کنیم. فرض کنید که $I_{i,p}$ ، i امین بازه Pr_p و τ یک شماره دنباله باشد که معرف یک ترتیب روی مقادیر اندازه‌گیری‌شده روی ویژگی Pr_p باشد. یک شاخص روی

بازه I که با $(In_{i,p})_\tau$ نشان داده می‌شود، یک نگاشت برای بازه $I_{i,p}$ در زمان τ به صورت زیر است:

$$(In_{i,p})_\tau : (Pr_p)_\tau \Rightarrow \begin{cases} true \text{ if } (Pr_p)_\tau \in I_{i,p} \\ false \text{ otherwise} \end{cases}$$

از این به بعد، هر زمان اندازه‌گیری از یک ویژگی را یک "رویداد" می‌نامیم. بنابراین، می‌گوییم که $(In_{i,p})_\tau$ برقرار است اگر امین "رویداد" Pr_p به بازه $I_{i,p}$ تعلق داشته باشد در غیراین صورت $(In_{i,p})_\tau$ برقرار نیست که به صورت $\neg (In_{i,p})_\tau$ نشان داده می‌شود. برای مثال، در سیستم حفاظت قطار، فرض کنید Pr_1 معرف ویژگی سرعت قطار باشد، در نتیجه، I_{21} معرف بازه Pr_1 خواهد بود و عبارت "$(In_{2,1})_3$ برقرار است" به این معنی است که سومین سرعت اندازه‌گیری شده قطار به دومین بازه Pr_1 تعلق دارد. توجه کنید که در هر زمان، فقط یک شاخص از یک ویژگی برقرار است، زیرا هر مقدار Pr_p به فقط یکی از بازه‌هایش تعلق دارد (رابطه ۶). به عبارت دیگر، این خصوصیت معرف این است که یک کمیت اندازه‌گیری شده در زمان τ نمی‌تواند به دو بازه از یک ویژگی متعلق باشد، زیرا بازه‌ها منفصل هستند.

$$\forall (I_{i,p}, I_{j,p}) : \nexists \tau, (In_{i,p})_\tau \text{ and } (In_{j,p})_\tau \text{ where } i \neq j \qquad (۶)$$

۲-۲- تعریف رخداد

یک رخداد اتفاق می‌افتد اگر تغییر مقدار یک ویژگی مانند Pr_p باعث شود تا Pr_p از بازه $I_{i,p}$ به بازه‌های مجاورش؛ یعنی $I_{i-1,p}$ یا $I_{i+1,p}$ گذر کند. به طور مشابه، تغییر در Pr_p از بازه‌های $I_{i-1,p}$ یا $I_{i+1,p}$ به بازه مجاورش $I_{i,p}$ معرف یک رخداد است. ما فرض می‌کنیم که تغییر مقدار یک ویژگی تدریجی است؛ به این معنی که تغییر مقدار یک ویژگی یا در داخل یک بازه قرار دارد یا این که به انتقال به بازه مجاور خودش منجر می‌شود. اکنون یک رخداد را به صورت رسمی تعریف می کنیم:

اگر دو رویداد متوالی از Pr_p که با τ و $\tau+1$ مشخص می‌شود، به بازه $I_{i,p}$ و یکی از بازه‌های مجاور آن؛ یعنی $I_{i-1,p}$ یا $I_{i+1,p}$ متعلّق باشد، می‌گوییم یک رخداد در زمان τ رخ داده است (رابطه‌های (۷) و (۸)). رابطه (۷) دو مورد را بیان می‌کند: (۱) آخرین زمانی که Pr_p در بازه $I_{i-1,p}$ یا $I_{i+1,p}$ بوده‌است، زمان τ است. این بدین معنی است که Pr_p در زمان τ در $In_{i,p}$ نبوده است که آن را با $(In_{i,p})_\tau$ نشان می‌دهیم. (۲) در زمان τ یک تغییر در مقدار Pr_p باعث شده است که Pr_p بازه $I_{i-1,p}$ یا $I_{i+1,p}$ را ترک کند و وارد بازه $I_{i,p}$ در زمان $\tau+1$ شود. به طور مشابه، رابطه (۸) بیان می‌کند که τ آخرین زمانی است که Pr_p در $I_{i,p}$ بوده است و در زمان جاری یک تغییر در مقدار Pr_p موجب شده‌است تا Pr_p بازه $I_{i,p}$ را ترک کند و وارد بازه $I_{i-1,p}$ یا $I_{i+1,p}$ در زمان $\tau+1$ شود. این بدین معنی است که Pr_p در $In_{i,p}$ در زمان $\tau+1$ نبوده است. بنابراین، ما این مورد را با $(In_{i,p})_{\tau+1}$ نشان می‌دهیم. همان طور که روابط (۷) و (۸) نشان می‌دهد، فقط از $In_{i,p}$ استفاده کرده‌ایم. این بدین معنی است که دغدغه ما تنها ورود به یا خروج ازبازه $In_{i,p}$ است و وارد شدن به یا خروج از بازه‌های مجاور $In_{i,p}$ دغدغه ما نیست. نمادهای "@T" معرف تغییر مقدار از false به true و "@F" معرف تغییر مقدار از true به false است که به ترتیب معرف قرارداشتن و قرارنداشتن در بازه $In_{i,p}$ است. رابطه های (۷) و (۸) معرف ورود به یا خروج از بازه $I_{i,p}$ در زمان $\tau+1$ است.

$$@T(In_{i,p})_\tau \stackrel{def}{\equiv} [(In_{i,p})_\tau \text{ and } (In_{i,p})_{\tau+1} \qquad (۷)$$

$$@F(In_{i,p})_\tau \stackrel{def}{\equiv} [(In_{i,p})_\tau \text{ and } (In_{i,p})_{\tau+1} \qquad (۸)$$

در حالتی که یک ویژگی از نوع منطقی باشد؛ یعنی مقادیر true و false داشته باشد، دو پارتیشن [true] و [false] خواهیم داشت. بنابراین، دو نوع رخداد می‌تواند وجود داشته باشد که به وسیله رابطه‌های (۹) و (۱۰) بیان بیان می‌شود. نمادهای "@T" و "@F" معرف بودن و نبودن در بازه $In_{1,1}$ است که $Pr_1=\{true,false\}$ و $I_{1,1}=[true]$ و $I_{2,1}=[false]$. رابطه‌های (۹) و (۱۰) می‌تواند از رابطه‌های (۷)و (۸) به دست آید. رابطه (۹) معرف خروج از بازه $I_{1,1}$ در زمان $t+1$ یا معرف ورود به بازه $I_{2,1}$ در زمان $\tau+1$ است و رابطه ۱۰ معرف ورود به بازه $I_{1,1}$ در زمان $\tau+1$ یا خروج از بازه $I_{2,1}$ در زمان $\tau+1$ است.

$$@T(In_{1,1})_\tau \stackrel{def}{\equiv} [\neg(In_{1,1})_\tau \text{ and } (In_{1,1})_{\tau+1}] \equiv [(In_{2,1})_\tau \text{ and } \neg(In_{2,1})_{\tau+1}] \quad (۹)$$

$$@F(In_{1,1})_\tau \stackrel{def}{\equiv} [(In_{1,1})_\tau \text{ and } \neg(In_{1,1})_{\tau+1}] \equiv [\neg(In_{2,1})_\tau \text{ and } (In_{2,1})_{\tau+1}] \quad (۱۰)$$

توجه داشته‌باشید که رخدادها می توانند همزمان اتفاق بیفتند. به عبارت دیگر، در هر مرحله زمانی بیش از یک رخداد می‌تواند رخ دهد. این بدین معنی است که مقدار ویژگی‌ها می‌تواند در هر زمان به صورت همزمان تغییرکند. در رابطه‌های (۱۱) تا (۱۴)، طرف چپ معرف تغییر مقدار Pr_p در زمان τ است و طرف راست معرف تغییر مقدار ویژگی دیگرمانند Pr_q در همان زمان است، اما همان طور که در رابطه (۶) بیان کردیم، حداکثر یک مقدار از یک ویژگی در هر مرحله از زمان می‌تواند تغییر کند.

$$@T(In_{i,p})_\tau \text{ and } @T(In_{k,q})_\tau \quad (۱۱)$$

$$@F(In_{i,p})_\tau \text{ and } @T(In_{k,q})_\tau \quad (۱۲)$$

$$@F(In_{i,p})_\tau \text{ and } @F(In_{k,q})_\tau \quad (۱۳)$$

$$I_{i,p} \in Pr_p,\ In_{k,q} \in Pr_q,\ p \neq q \quad (۱۴)$$

۲-۳- تعریف ثابت

فرض کنید τ و $\tau+1$ دو رویداد متوالی از ویژگی Pr_p باشد. می‌گوییم یک *ثابت* روی بازه $I_{j,q}$ وجود دارد اگر: (۱) هر دو رویداد متعلّق به بازه $I_{j,q}$ باشد یا (۲) هیچ یک از آن دو در بازه $I_{j,q}$ نباشند. حالت اوّل را یک ثابت برقرار رابطه (۱۵) و حالت دوّم را یک ثابت نابرقرار گوییم رابطه (۱۶). رابطه (۱۵) اظهار می‌دارد که $In_{j,q}$ در زمان‌های τ و $\tau+1$ برقرار بوده است و رابطه (۱۶) اظهار می‌دارد که $In_{j,q}$ در زمان‌های τ و $\tau+1$ برقرار نبوده است.

$$t(In_{j,q})_{\tau,\tau+1} \stackrel{def}{\equiv} [(In_{j,q})_\tau \text{ and } (In_{j,q})_{\tau+1}] \quad (۱۵)$$

$$f(In_{j,q})_{\tau,\tau+1} \stackrel{def}{\equiv} [\neg(In_{j,q})_\tau \text{ and } \neg(In_{j,q})_{\tau+1}] \quad (۱۶)$$

مشابه با رخدادها، ثابت‌ها می‌توانند همروند باشند؛ یعنی ترکیبی از ثابت‌های برقرار و نابرقرار در یک زمان وجود داشته باشد مشروط بر این که هر ثابت متعلّق به یک ویژگی جداگانه باشد. ثابت‌های همروند را با $\Sigma_{i,r}\, t(In_{i,r})_{\tau,\tau+1}$ یا $\Sigma_{i,r}\, f(In_{i,r})_{\tau,\tau+1}$ نشان می‌دهیم. این نمادها، ثابت‌ها را روی بازه‌های جداگانه‌ای از ویژگی‌ها توصیف می‌کند. بنابراین برای هر دو ثابت همروند زیر شرط $p \neq q$ وجود دارد:

$$(In_{i,p})_{\tau,\tau+1} \ ,\ (In_{j,q})_{\tau,\tau+1}$$

۲-۴- نیاز ایمنی

همان طور که در بخش‌های ۲ و ۳ بیان کردیم، یک نیاز ایمنی معرف این است که یک رخداد بد که معرف نقض نیازکاربر سیستم است، رخ نمی‌دهد. ما این نقض را به صورت ترکیب رخدادها و ثابت‌ها نشان می‌دهیم. بروز چندین رخداد را با:

$$\Sigma_{i\geq 1} @T(In_{i,p})_\tau \text{ یا } \Sigma_{i\geq 1} @F(In_{i,p})_\tau$$

و چندین ثابت را با:

$$\Sigma_{i\geq 0}\, t(In_{j,r})_\tau \text{ یا } \Sigma_{i\geq 0}\, f(In_{j,r})_\tau$$

نشان می‌دهیم. رخداد را با رابطه‌های (۷) و (۸) و ثابت را با رابطه‌های (۱۵) و (۱۶) تعریف کردیم. فرض کنید که ویژگی Pr_2 در سیستم حفاظت قطار معرف ناحیه‌ای از خط آهن باشد که اوّلین بازه آن، بازه "هشدار" است، بنابراین $In_{1,2}$ معرف "ناحیه هشدار" است و فرض کنید که ویژگی Pr_3 معرف "ترمز قطار" باشد که اولین بازه‌اش، بازه "آزاد" است. بنابراین $In_{1,3}$ معرف "ترمز آزاد" است. درنتیجه رخداد $@T(In_{1,2})_\tau$ که به معنی $[\neg(In_{1,2})_\tau \text{ and } (In_{1,2})_{\tau+1}]$ همراه با ثابت $t(In_{1,3})_{\tau,\tau+1}$ که به معنی $[(In_{1,3})_\tau \text{ and } (In_{1,3})_{\tau+1}]$ است، یک نقض را نشان می‌دهد. ثابت $t(In_{1,3})_{\tau,\tau+1}$ اظهار می دارد که در زمان رخداد (یعنی ورود قطار به ناحیه هشدار) ترمز قطار آزاد بوده است.

۳- توصیف اتوماتا

مدل‌سازی رسمی رفتار نرم‌افزار و ویژگی‌های سیستم با استفاده از اتوماتا به عنوان پایه‌ای برای پایش حین اجرا، ایده‌ای است که در ادبیات وارسی حین اجرا بیان شده است [۵]. در این میان، شبکه‌های پتری اتوماتاهای مناسبی برای مدل‌سازی و تحلیل سیستم‌های رخداد گسسته همروند و غیرهمگام[۶] هستند [۶]. علاقه‌مندی قابل ملاحظه‌ای به شبکه‌های پتری برای مطالعه در حوزه مسائل کنترل

سرپرستی نشان داده شده است [۱۲و۱۱و۱۰و۹ و۸ و۷] زیرا این شبکه‌ها قدرت مدل‌سازی و تحلیل خوبی دارد. به علاوه، شبکه‌های سطح بالا مانند شبکه‌های پتری رنگی[۷] توانایی‌های اضافه‌ای در مدل‌سازی سیستم‌ها دارند [۱۴]. در این بخش، ما می‌خواهیم روشی برای نگاشت توصیف رخدادها و ثابت‌هایی که در بخش ۲ بیان کردیم، به مؤلفه‌های شبکه پتری ارائه دهیم. شبکه‌های پتری حاصل رفتار سیستم و محیط آن را نشان می‌دهد.

تعریف: یک شبکه پتری معمولی یک ۴ تایی به صورت $PN = (P, T, F, m_0)$ است که در آن $P = \{P_1, P_2, \ldots, P_m\}$ یک مجموعه متناهی از مکان‌هاست و با حباب نشان داده می‌شود و $T = \{T_1, T_2, \ldots, T_n\}$ مجموعه‌ای متناهی از گذارهاست که با خط عمودی پررنگ نشان داده می‌شود و $F = \{F_1, F_2, \ldots, F_k\}$ یک مجموعه متناهی از قوس‌های جهت‌دار از مکان‌ها به گذارها یا بالعکس است که با پیکان نمایش داده شده است و $M: P \rightarrow N$ یک نشانه‌گذاری[۸] است که معرف تعداد نشانه‌ها[۹] در مکان‌هاست و m_0 معرف نشانه‌گذاری اولیه است [۶]. یک نشانه‌گذاری با برداری مشخص می‌شود که اعضایش معرف تعداد نشانه‌ها در مکان‌هاست. برای مثال، بردار $m_0=[1,0,0]$ معرف این است که $m_0(P_1)=1, m_0(P_2)=0, m_0(P_3)=0$ یعنی مکان‌های P_1، P_2 و P_3 در ابتدا به ترتیب شامل یک، صفر و صفر نشانه است. مشابه با بردار m_0 ، بردار m_i معرف نشانه‌گذاری شبکه پتری در زمان i ام است.

تعریف: فرض کنید که مجموعه همه مکان‌های ورودی به خروجی از گذار T_i به ترتیب با $\bullet T_i$ و $T_i \bullet$ نشان داده شوند. گذار T_i از مجموعه گذارهای T در شبکه پتری $PN = (P, T, F, M)$ را در نشانه‌گذاری m_i توانا[۱۰] گوییم اگر $\forall p \in \bullet T_i, m_i(p) \geq 1$. اگر گذار T_i توانا باشد، می‌تواند عمل کند[۱۱]. عمل کردن گذار T_i در نشانه گذاری $m_j(p)$ که با رابطه ۱۷ نشان داده می‌شود که به نشانه‌گذاری $m_{j+1}(p)$ منجر می‌شود. در این حالت می‌گوییم که $m_{j+1}(p)$ از $m_j(p)$ قابل دسترسی است.

$$m_{j+1}(p) = \begin{cases} m_j(p) - 1, & p \in T_i \bullet \\ m_j(p) + 1, & p \in \bullet T_i \end{cases} \qquad (۱۷)$$

۳-۱- ساختن اتوماتا

وقتی مکان‌ها، گذارها و نشانه‌گذاری‌ها تعریف شدند، از ویژگی‌های محیطی سیستم برای ساختن اتوماتای پتری استفاده می‌کنیم. برای هر ویژگی مانند p یک زیرشبکه درنظر می‌گیریم که در آن برای هر $I_{i,p}$، یک محل منظور می‌شود و محل دارای نشانه، متناظر با بازه جاری از ویژگی pاست. حال ثابت‌های $t(In_{i,r})_\tau$ یا $f(In_{i,r})_\tau$ و رخدادهای $F(In_{i,p})_\tau$ یا $@T(In_{i,p})_\tau$ را که در بخش ۲ توصیف شدند، به مکان‌ها و گذار بین این مکان‌ها منتسب می کنیم.

ویژگی‌های محیطی دو نوع هستند: پایشی و کنترلی. یک ویژگی پایشی مانند "ناحیه ورود قطار" ویژگیی است که باید به وسیله سیستم پایش شود و یک ویژگی کنترلی مانند "ترمز قطار" ویژگیی است که باید برای کنترل محیط مانند قطار استفاده شود. پس از آن که زیرشبکه‌ها برای ویژگی‌ها مشخص شدند، یک شبکه سراسری از اتصال زیرشبکه‌های حاصل به دست می‌آوریم. زیرشبکه‌ها، شبکه‌های پتری هستند که برای ویژگی‌های پایشی و کنترلی ساخته شده‌اند و شبکه سراسری، در حقیقت اتصال بین این دو نوع ویژگی‌های متناظر را نشان می‌دهد. برای مثال، در سیستم حفاظت قطار، ویژگی‌های "سرعت قطار" و "ناحیه ورود قطار" که ویژگی‌های پایشی هستند، برای تعیین وضعیت "ترمز قطار" که ویژگی کنترلی است، استفاده می‌شوند. بنابراین، زیرشبکه‌های پتری حاصل از ویژگی‌های پایشی به زیرشبکه‌های پتری حاصل از ویژگی‌های کنترلی متصل می‌شوند. سپس شبکه پتری سراسری حاصل یک توصیف مبتنی بر حالت از رفتار محیط سیستم و پاسخ‌های سیستم را مشخص می‌کند.

علاوه بر شبکه سراسری پتری، شبکه‌ای نیز برای نقض نیازهای کاربران سیستم ساخته می‌شود. عمل کردن هر گذار در این شبکه با یک نشانه‌گذاری قابل دسترس رابطه (۱۷) نشان داده می‌شود که به معنی نقض یک نیاز است. این نقض نیاز کاربر، به خاطر بروز یک رخداد و برقراری یک یا چند ثابت است. وقتی رابطه ۱۷ برای نقض یک نیاز استفاده می‌شود، نماد $m_j(p)$ یک حالت (نشانه‌گذاری) امن

و نماد $m_{j+1}(p)$ یک حالت (نشانه‌گذاری) ناامن یا بحرانی را نشان می‌دهد. عبارت $@T(In_{i,p})_{\tau}$ به این معنی است که اگر گذار عمل کند، مقدار $In_{i,p}$ در مکان ورودی شبکه (که یک مکان امن است) false است و در مکان خروجی (که یک محل ناامن یا بحرانی است) برقرار می شود (true). به طور مشابه، عبارت $@F(In_{i,p})_{\tau}$ به این معنی است که مقدار $In_{i,p}$ در مکان ورودی شبکه برقرار است (true) و در مکان خروجی شبکه برقرار نیست (false). عبارت $t(In_{j,r})_{\tau,\tau+1}$ به این معنی است که مقدار $In_{j,r}$ در مکان ورودی برقرار است(true) و وقتی گذار عمل می‌کند، همچنان برقرار باقی می ماند. به طور مشابه عبارت $f(In_{j,r})_{\tau,\tau+1}$ به این معنی است که مقدار $In_{j,r}$ در هردو مکان برقرار نیست. چنین توصیفی نشان دهنده حالت‌های ناامن یا بحرانی (یعنی نقض یک نیاز کاربر) است که برای وارسی حین اجرای عملیات سیستم استفاده می‌شود. اگر یک مکان در شبکه پتری را یک حالت حین اجرا در نظر بگیریم، شبکه پتری پلی روی شکاف بین رخدادهای محیطی و عملیات حین اجرای سیستم خواهد بود که برای وارسی حین اجرای رفتار سیستم به منظور مطابقت با نیازهای ایمنی مبتنی بر دامنه و سطح بالای کاربران استفاده می شود.

۴- طراحی شبیه‌ساز پایشگر

هنگامی که یک رخداد در محیط سیستم اتفاق می‌افتد، پایشگر، نشانه‌گذاری جاری در شبکه پتری نقض نیازها که یک نشانه‌گذاری امن است و همچنین، رخداد و ثابت‌هایی را که معرف شرایط محیط هستند، می‌گیرد. اگر گذاری از شبکه پتری حاصل بتواند عمل شود، نشان دهنده این است که پاسخ سیستم نامطلوب بوده و یک نقض رخ‌داده است. برای مثال، در سیستم حفاظت قطار که قطار وارد ناحیه هشدار می‌شود و سرعت بالا دارد، مشخص می‌شود که سیستم سرعت قطار را کنترل نکرده است و نقض یک نیاز کاربر محسوب می‌شود. این نقض با عمل‌کردن یک گذار در شبکه پتری کشف می‌شود. اگر سرعت قطار پایین باشد، هیچ گذاری عمل نمی‌کند و در نتیجه هیچ نقض نیازی نداریم.

همان طور که در بخش ۲-۲ بیان شد، یک رخداد به عنوان تغییر مقدار یک ویژگی تعریف می‌شود که به ایجاد یک گذار از بازه جاری ویژگی به بازه بعدی یا قبلی آن منجر می‌شود. بنابراین، برای تعیین این که آیا اتفاقی رخ داده است یا نه، پایشگر مقدار تغییریافته یک ویژگی را از محیط سیستم دریافت می‌کند و آن را با می‌نیمم مقدار هر بازه از آن ویژگی مقایسه می‌کند تا بازه متناظر در ویژگی را تعیین کند. پس از آن، پایشگر پاسخ سیستم را با پاسخ صحیح در بازه متناظر مقایسه می‌کند. متوسط زمان مقایسه از مرتبه $n/2$ است که n تعداد بازه‌های ویژگی است.

برای شبیه‌سازی پایشگر در کنترل سرپرستی، مکانیسمی برای اداره رخدادهای همروند لازم است، زیرا پایشگر باید قادر به دریافت و اداره رخدادهایی که به صورت همروند در محیط سیستم رخ می‌دهد، باشد. به طور سنتی، الگوی برنامه‌نویسی چندنخی برای پشتیبانی همروندی در رایانه‌های تک‌هسته استفاده می‌شود، اما رشد فزاینده پردازنده‌های چندهسته، الگوهای جدید همروندی را در استفاده از پردازنده‌ها ایجاد کرده‌است. علاوه براین، پردازنده‌های چندهسته می‌توانند برای افزایش کارآیی اجرای برنامه ها در اداره داده‌ها و وظایف موازی، مدیریت شوند. کتابخانه موازی وظایف (TPL) کتابخانه‌ای در محیط NET. است که برنامه‌نویسان را برای استفاده از پردازنده‌های چندهسته در اجرای موازی برنامه‌ها، توانا می‌سازد. در TPL، یک وظیفه به یک نخ منتسب می شود و سپس وظایف موازی به صورت نخ‌های همروند اجرا می‌شوند. TPL مجموعه‌ای از واسط‌های برنامه‌نویسی کاربردی را فراهم می‌سازد که به وسیله آنها برنامه‌نویسان قادر می‌شوند عملیاتی مانند انتظار برای اتمام وظایف، لغو وظایف، ایجاد وظایف طولانی اجرا و اجرای وظایف را به صورت همگام اجرا کنند. علاوه براین، آن الگوی زمانبندی وظایف را با تعیین سطح همروندی و اولویت‌ها، پیکربندی می‌کند. پایشگر را با استفاده از مکانیسم‌های: (۱) نخ، (۲)

وظیفه (۳) ترکیب نخ و وظیفه شبیه‌سازی می‌کنیم و آن را در "سیستم حفاظت قطار" در بخش بعدی به کار می‌بریم تا کارآیی مکانیسم‌ها را با یکدیگرمقایسه‌کنیم.

۵- مورد مطالعه: سیستم حفاظت قطار

در این بخش روش پیشنهادی را برای طراحی پایشگر کنترل سرپرستی سیستم حفاظت قطار به کار می‌بریم و به بررسی کارایی آن می‌پردازیم. این سیستم را Cab signaling گویند که به صورت پیوسته اطلاعاتی را درباره محل و سرعت قطار دریافت می‌کند تا آن را با اتخاذ کنش‌های مناسب روی سرعت و ترمز کنترل کند. این سیستم در بسیاری از کشورها مانند فرانسه، آلمان و ژاپن استفاده می‌شود. این سیستم سیگنال‌های سبز، زرد یا قرمز را برای راننده قطار نمایش می‌دهد تا با توجه به آن راننده قطار سرعت خود را تنظیم کند. سیگنال‌های سبز، زرد و قرمز به ترتیب معرف نواحی "امن"، "محدود" و "خطرناک" است. در ناحیه امن قطار می‌تواند دارای هر سرعتی باشد، در ناحیه محدود، سرعت قطار دارای حداکثر مجازی است و در ناحیه سوم، خطر تصادف وجود دارد و سرعت باید حداقل شود. وقتی که راننده قطار به سگینال صادره توجهی ندارد و از محدوده سرعت مجاز تخطی می‌کند، این سیستم وظیفه دارد تا به راننده قطار هشدار دهد. اگر راننده قطار به هشدارها پاسخ ندهد، سیستم خود برای عمل ترمزگیری اقدام خواهدکرد. سرعت و ترمز قطار براساس نوع سیگنال مشخص می‌شود. کنش‌هایی که انتظار می‌رود تا به وسیله کنترل سرپرستی سیستم اتخاذ شود، به قرار زیر است:

الف- اگر سرعت قطار زیر سرعت مجاز است، کنشی لازم نیست؛

ب- اگر قطار با سرعت خیلی زیاد در هنگام اعلام سیگنال قرمز حرکت می‌کند، سیستم باید هشدار دهد و سپس اگر راننده قطار هشدار را جدی نگرفت، خود ترمز قطار را بکشد؛

ج- سیگنال زرد معرف این است که قطار باید آماده توقف حرکت در سیگنال بعدی باشد. بنابراین، کنترل کننده سرپرستی باید منحنی ترمز را محاسبه کند تا سرعت قطار را در هنگام ورود به ناحیه متعادل کاهش دهد؛

د- هنگامی که قطار متوقف شد، ترمزهایش باید آزاد شود.

در زیر بخش‌های بعدی، پایشگر را با استفاده از موارد زیر شبیه‌سازی می‌کنیم: (۱) اتوماتا؛ (۲) تعیین ویژگی‌های قطار به عنوان محیط سیستم و (۳) توصیف رخدادها، ثابت‌ها و نیازهای ایمنی کاربران.

۵-۱ – مدل سازی داده‌ها

با در نظرگرفتن کنش‌هایی که باید به وسیله کنترل سرپرستی سیستم برای کنترل محیط سیستم گرفته شود، سه ویژگی محیطی برای کنترل‌کردن محیط استفاده می‌شود: (۱) سرعت قطار که با نماد Pr_1 نشان می‌دهیم؛ (۲) ترمز قطار که با نماد Pr_2 نشان می‌دهیم و (۳) ناحیه‌ای که قطار وارد می‌شود که با نماد Pr_3 نشان می‌دهیم. اولین و سومین ویژگی از نوع شمارشی (ر. ک. بخش ۲-۱) و دومین ویژگی یک عدد صحیح است. سیستم نواحی را که قطار وارد می‌شود، پایش می‌کند و بر اساس آن قطار را از طریق سرعت و ترمز آن کنترل می‌کند. محاسبات برای اعمال ترمز و کاهش سرعت قطار به وسیله نرم‌افزار سیستم انجام می‌شود.

Pr_1: VELOCITY = $'^{+}$ = $\{v_1, v_2, \ldots, v_n\}$ $v_i < v_j$ for i < j

Pr_2: BRAKE = {putOn, putOff}

Pr_3: REGION = {free, moderate, cautionary}

براساس بخش ۲-۱ ، ویژگی Pr_1 را به بازه‌های جدا از هم و متناهی با رابطه ترتیب "<" تقسیم می‌کنیم؛ به طوری که برای هر دو بازه متوالی، مانند $[a_1 \ldots a_j]$ و $[a_{j+1} \ldots a_m)$ سوپرمم اولین بازه و می‌نیمم دومین بازه است. براساس نظر خبره سیستم، سرعت کمتر از یک مقدار مشخص، مانند v_i معرف سرعت مجاز در همه وضعیت‌هاست و سرعت بین دو مقدار مشخص مانند v_i و v_j وقتی مجاز است که قطار سیگنال قرمزی را دریافت نکند و سرعت بیش از مقدار v_j

وقتی مجاز است که قطار از سیستم سیگنال سبز دریافت کند. در نتیجه، ویژگی Pr_1 به سه بازه $I_{1,1}$، $I_{2,1}$ و $I_{3,1}$ تقسیم می‌شود که:

$$\text{VELOCITY} = r^+ = \{v_1, v_2, \ldots, v_n\} = I_{1,1} \cup I_{2,1} \cup I_{3,1} :$$

$$I_{1,1} = [v_1 \ldots v_i), I_{2,1} = [v_i \ldots v_j), I_{3,1} = [v_j \ldots v_n)$$

پس از تعیین بازه‌ها، شاخص‌ها را تعریف می‌کنیم. فرض کنید که بازه $I_{i,1}$، iامین بازه ویژگی Pr_1 و τ یک شماره توالی باشد که معرف یک ترتیب روی مقادیر اندازه‌گیری شده ویژگی Pr_1 باشد. یک شاخص روی بازه I که با $(In_{i,1})_\tau$ نشان داده می‌شود، یک نگاشت برای بازه $I_{i,1}$ در زمان τ است (رابطه ۱۸).

$$(In_{i,1})_\tau : (Pr_1)_\tau = \begin{cases} true; \text{ when } (Pr_1)_\tau \in I_{i,1} \\ false; \text{ otherwise} \end{cases} \quad (۱۸)$$

شاخص‌های $In_{1,1}$، $In_{2,1}$ و $In_{3,1}$ معرف نمادهای "کم"، "متوسط" و "زیاد" هستند. ویژگی Pr_2 (برای ترمز) شامل دو بازه [putOn] و[putOff] با مقادیر منطقی و ویژگی Pr_3 (برای ناحیه) شامل سه بازه free، moderate و cautionary با مقادیر شمارشی است. در حقیقت، هر بازه یک کمیت است که نشان دهندهٔ مجموعه‌ای از مقادیر است. این کمیت‌ها به عنوان شاخص‌های بازه‌ای منظور می‌شوند:

$$In_{1,2} = \text{putOn}, \; In_{2,2} = \text{putOff}$$
$$In_{1,3} = \text{free}, \; In_{2,3} = \text{moderate}, \; In_{3,3} = \text{cautionary}$$

همان طور که در بخش ۴ بیان شد، هنگامی که مقدار یک ویژگی محیطی تغییر می‌کند، پایشگر آن را از محیط سیستم دریافت می‌کند و آن را با مینیمم مقدار هر بازه از ویژگی مقایسه می‌کند تا بازه متناظر با آن را تعیین کند. با تعیین بازه، کنش مورد انتظار سیستم مشخص می‌شود.سپس پایشگر کنش مورد انتظار را با پاسخ واقعی سیستم مقایسه می‌کند که درستی رفتار سیستم مشخص شود.

۵-۲- تعریف رخداد و ثابت

براساس رابطه‌های (۷) و (۸) (یک رخداد به معنی تغییر بازه در یک خصیصه است) و رابطه‌های (۱۵) و (۱۶) (یک ثابت به معنی عدم تغییر بازه یک ویژگی در هنگام یک رخداد است)، ما رخدادها و ثابت‌های محیطی سیستم حفاظت قطار را تعریف می‌کنیم. در ارتباط با شاخص‌های بازه‌ای برای ویژگی Pr_3 (ناحیه آزاد، محدود و هشدار) رخدادها و ثابت‌های زیر را می‌توان داشت:

$T(free)_\tau$, $@F(free)_\tau$, $@T(moderate)_\tau$,$F(moderate)_\tau$, $@T(cautionary)_\tau$ and $@F(cautionary)_\tau$

$t(free)_{\tau,\tau+1}$, $f(free)_{\tau,\tau+1}$, $t(moderate)_{\tau,\tau+1}$,$f(moderate)_{\tau,\tau+1}$, $t(cautionary)_{\tau,\tau+1}$ and $f(cautionary)_{\tau,\tau+1}$

توجه داشته باشید که رخداد "@F" می‌تواند با رخداد "@T" روی بازه دیگر تعریف شود. برای مثال، رخداد "@F(*moderate*)" زمانی رخ می‌دهد که "@T(*free*)$_\tau$" یا "@T(*cautionary*)$_\tau$" رخ دهد. به طور مشابه یک ثابت از نوع "t" می‌تواند با دو ثابت "f" بیان شود. برای مثال، ثابت "t(*free*)$_{\tau,\tau+1}$" می‌تواند با ثابت‌های "f(*moderate*)$_{\tau,\tau+1}$" و "f(*cautionary*)$_{\tau,\tau+1}$" بیان شود. به طور مشابه، رخدادها و ثابت‌ها برای ویژگی‌های Pr_1 و Pr_2 تعریف می‌شوند.

۵-۳- نیازهای ایمنی

با استفاده از تعریف نیاز ایمنی (بخش ۲-۴ را ملاحظه کنید) و کنش‌هایی که باید به وسیله سیستم در برابر رخدادهای محیطی انجام شود (این کنش‌ها در بخش ۵ با موارد الف تا د مشخص شدند)، نیازهای ایمنی را برای سیستم حفاظت قطار تعیین می‌کنیم. در ارتباط با کنش "ب" در بخش ۵، یک رخداد بد وجود دارد که باید به وسیله پایشگر به صورت زیر هشدار داده شود. هنگامی که قطار می‌خواهد وارد ناحیه هشدار شود، سیستم باید یک سیگنال قرمز را برای قطار صادرکند. بنابراین، یک رخداد بد اتفاق خواهد افتاد اگر سیستم چنین سیگنالی را صادرنکند یا اینکه راننده قطار توجهی به آن نکند. در این راستا، نقض‌های نیازهای ایمنی عبارتند از: (۱) قطار با سرعت بالا در ناحیه هشدار حرکت می‌کند و (۲) ترمز قطار در ناحیه هشدار آزاد است. این دو نقض را با SRV_1 و SRV_2 تعریف می‌کنیم و با رابطه‌های (۱۹) و (۲۰) نشان می‌دهیم.

$$SRV_1: @T(cautionary)_\tau \text{ and } t(high)_{\tau,\tau+1} \quad (۱۹)$$

$$SRV_2: @T(cautionary)_\tau \text{ and } t(putOff)_{\tau,\tau+1} \quad (۲۰)$$

در ارتباط با کنش "ج" در بخش ۵، یک رخداد بد وجود دارد که باید به وسیله پایشگر به شیوه زیر هشدار داده شود: هنگامی که قطار می‌خواهد در سیگنال بعدی توقف کند، سیستم باید یک سیگنال زرد را صادر کند. بنابراین، یک رخداد بد رخ خواهد داد وقتی که سیستم،

سیگنالی را صادر نکند (این می‌تواند به دلیل عدم محاسبه صحیح منحنی ترمز کاهش سرعت قطار باشد) یا این که راننده قطار، سرعت قطار را کاهش ندهد. بنابراین، نقض‌های نیازهای ایمنی که با SRV_3 و SRV_4 (رابطه‌های ۲۱ و ۲۲) نشان داده می‌شوند، عبارتند: از (۱) سرعت قطار در هنگام ورود به ناحیه محدود بالاست و کاهش نمی‌یابد و (۲) وقتی قطار وارد ناحیه محدود می‌شود، راننده قطار ترمز نمی‌گیرد.

SRV_3: $@T(moderate)_\tau$ and $t(high)_{\tau,\tau+1}$ and $\neg@F(high)_\tau$ (۲۱)

SRV_4: $@T(moderate)_\tau$ and $@F(putOn)_{\tau,\tau+1}$ (۲۲)

نقض‌های SRV_1 و SRV_2 بحرانی و نقض‌های SRV_3 و SRV_4 ناامن هستند. به ترتیب نقض‌های بحرانی و ناامن هستند. یک نقض بحرانی معرف یک موقعیت خیلی خطرناک است، در حالی که یک نقض ناامن معرف موقعیتی است که می‌تواند به وضعیت خطرناک منجر شود.

در ارتباط با کنش "الف" در بخش ۵، هیچ نیاز ایمنی وجود ندارد، زیرا هیچ رخداد بدی اتفاق نمی‌افتد حتی اگر سیستم سیگنال قرمزی را صادر کند یا این که دستور ترمز گرفتن بدهد و به طور مشابه، در ارتباط با کنش "د" در بخش ۵، هیچ نیاز ایمنی وجود ندارد، زیرا اگر ترمز آزاد نشود، رخداد بدی اتفاق نمی‌افتد.

۵-۴- ساخت اتوماتا

براساس بخش ۳-۱، دو اتوماتا ساخته می‌شود: (۱) برای نمایش رفتار سیستم و (۲) برای نمایش نقض‌های بحرانی و ناامن نیازهای ایمنی. از اتوماتای اول برای پایش رفتار سیستم و از اتوماتای دوم برای کنترل سیستم استفاده می‌شود. برای ساختن اتوماتای اول، یک شبکه برای هر ویژگی محیطی ساخته می‌شود و سپس یک شبکه سراسری با استفاده از اتصال هر شبکه ایجاد می شود. براساس بخش ۵-۱، محیط سیستم یعنی قطار، ۳ ویژگی Pr_1، Pr_2 و Pr_3 دارد که به ترتیب شامل ۳، ۲ و ۳ بازه است. بنابراین، سه شبکه پتری برای ویژگی‌های Pr_1، Pr_2 و Pr_3 ساخته می‌شود (نک. بخش ۳-۱). در زیر، مجموعه‌های P، T و m0 تا m2 به ترتیب مکان‌ها، گذارها و نشانه‌گذاری‌ها را برای شبکه اول توصیف می‌کند که محل‌های P_0 تا P_2 معرف شاخص‌های بازه‌ای "کم"، "متوسط" و "زیاد" برای سرعت هستند و نشانه‌گذاری‌های m_1 و m_2 از نشانه‌گذاری m_0 بر طبق رابطه ۱۷ ایجاد می‌شوند.

$P = \{P_0, P_1, P_2\}$, $T = \{T_0, T_1, T_2, T_3\}$, $\bullet T_0=\{P_0\}$, $T_0\bullet=\{P_1\}$, $\bullet T_1=\{P_1\}$, $T_1\bullet=\{P_2\}$, $\bullet T_2=\{P_1\}$, $T_2\bullet=\{P_0\}$, $\bullet T_3=\{P_2\}$, $T_3\bullet=\{P_1\}$, $m_0=\{1,0,0\}$, $m_1=\{0,1,0\}$, $m_2=\{0,0,1\}$

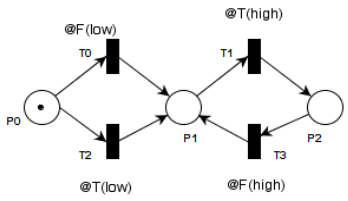
به طور مشابه، مجموعه‌های P´، T´ و m'_0 و $m'1$ به ترتیب مکان‌ها، گذارها و نشانه‌گذاری‌ها را برای شبکه دوم توصیف می‌کند که محل‌های P'_1 و P'2 به ترتیب معرف شاخص‌های بازه‌های "ترمز گرفته‌شده" و "ترمز آزاد" و نشانه‌گذاری از نشانه‌گذاری بر طبق رابطه (۱۷)، ایجاد می‌شود.

$P' = \{P'_1, P'_2\}$, $T' = \{T_4, T_5\}$, $\bullet T_4=\{P'_1\}$, $T_4\bullet=\{P'_2\}$, $\bullet T_5=\{P'_1\}$, $T_5\bullet=\{P'_2\}$, $m'_0=\{1,0\}$, $m'_1=\{0,1\}$

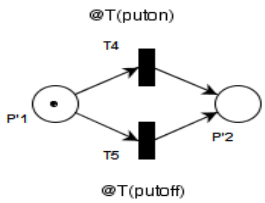
به طور مشابه، مجموعه‌های P´، T´ و m'_0 و m'_2 به ترتیب مکان ها، گذارها و نشانه‌گذاری‌ها را برای سومین شبکه توصیف می‌کند که محل‌های P''_1 ، P''_2 و P''_3 معرف شاخص بازه‌های "آزاد"، "محدود" و "هشدار" است و نشانه‌گذاری‌های m'_1 و m'_2 از نشانه‌گذاری m'_0 براساس رابطه (۱۷) ایجاد می‌شود.

$P'' = \{P''_1, P''_2, P''_3\}$, $T'' = \{T_6, T_7, T_8, T_9\}$, $\bullet T_6=\{P''_1\}$, $T_6\bullet=\{P''_2\}$, $\bullet T_7=\{P''_2\}$, $T_7\bullet=\{P''_1\}$, $\bullet T_8=\{P''_2\}$, $T_8\bullet=\{P''_3\}$, $\bullet T_9=\{P''_3\}$, $T_9\bullet=\{P''_2\}$, $m''_0=\{1,0,0\}$, $m''_1=\{0,1,0\}$, $m''_2=\{0,0,1\}$

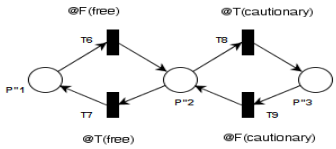
شکل های (۱)،(۲) و (۳) به ترتیب شبکه‌های پتری را برای سه شبکه فوق‌الذکر نشان می‌دهد که مکان های نشانه‌دار معرف مکان‌های جاری در هر شبکه است.



شکل (۱): شبکه پتری برای ویژگی Pr_1



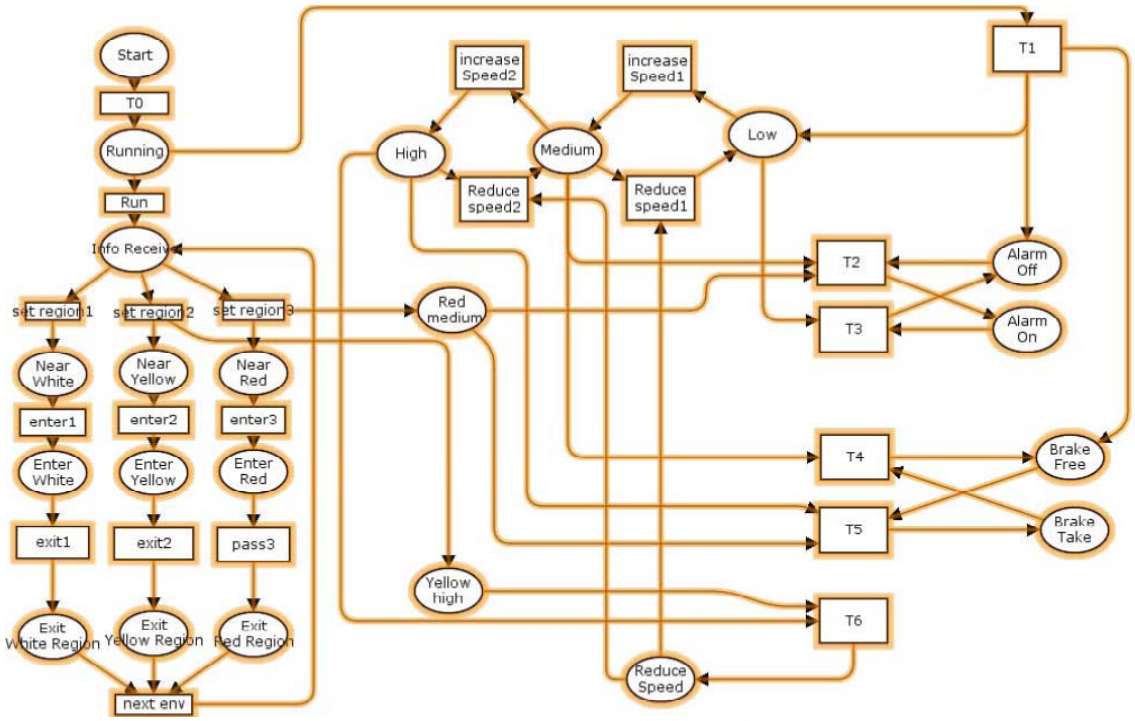
شکل (۲): شبکه پتری برای ویژگی Pr_2



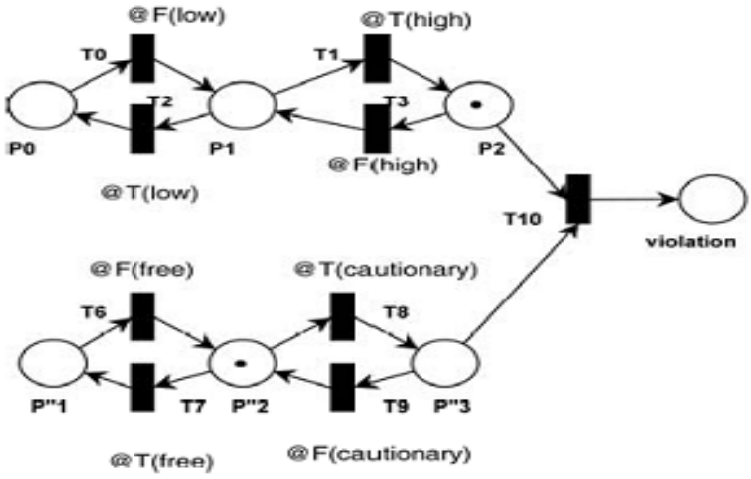
شکل (۳): شبکه پتری برای ویژگی Pr_3

ویژگی Pr_3 که معرف ناحیه(REGION) است، یک ویژگی پایشی است، زیرا به وسیله سیستم نظارت می‌شود، در حالی که ویژگی‌های Pr_1 و Pr_2 که به ترتیب معرف سرعت (VELOCITY) و ترمز(BRAKE) است، نقش ویژگی‌های هم پایشی و هم‌کنترلی را ایفا می‌کنند زیرا هر دو ویژگی به وسیله سیستم پایش می‌شوند، تا اگر لازم باشد تنظیم (کنترل) می‌شوند. شکل (۴)، شبکه سراسری است که با استفاده از زیر شبکه‌ها ساخته شده است. در این شکل، زیرشبکه مربوط به ویژگیPr_3 (سمت چپ شکل) ورودی به سیستم است، در حالی که زیر شبکه‌های مربوط به ویژگی‌های Pr_1 و Pr_2 (بالا و سمت راست شکل) هم ورودی و هم خروجی سیستم است.

اکنون به روش ساخت اتومای نقض نیازها می پردازیم که نشان دهنده نقض‌های بحرانی و ناامن نیازهای ایمنی است و با توجه به رابطه‌های (۱۹) تا (۲۳) به دست می‌آید. برای این که رخداد $@T(cautionary)_\tau$ در رابطه (۱۹) عمل کند، گذار T_8 در شکل (۳) باید توانا باشد و ثابت $t(high)_{\tau,\tau+1}$ در رابطه (۱۹) معرف این است که محل P_2 در شکل ۱، یک نشانه دارد. رخداد $@T(cautionary)_\tau$ به همراه ثابت $t(high)_{\tau,\tau+1}$. در رابطه (۱۹) با شکل(۵)نشان داده می‌شود که در آن نشانه‌گذاری $m_0=[0, 0, 1, 0, 1, 0]$، نشانه‌گذاری اولیه است. هنگامی‌که گذار T_8 در شکل (۵) عمل می‌کند، نشانه‌ها از محل‌های P_2 و P''_2 حذف می‌شود و محل P''_3 دارای نشانه می‌شود که با نشانه‌گذاری $m_1=[0, 0, 0, 0, 0, 1]$ مشخص می‌شود. نشانه گذاری m_1 از نشانه‌گذاری m_0 به وسیله پایشگر با استفاده از رابطه (۱۷) به دست می‌آید. اتوماتای نقض برای رابطه‌های (۲۰) تا (۲۲) به صورت مشابه ساخته می‌شود.



شکل (۴): شبکه پتری سیستم حفاظت قطار



شکل (۵): اتوماتای نقض نیاز SRV_1

۵-۵- شبیه‌سازی پایشگر

ما از زبان C# همراه با نخ‌ها و از کتابخانه TPL در محیط NET. برای : (۱)شبیه‌سازی رخدادهای همروند و ثابت‌ها در رفتار سیستم و (۲) مشاهده و وارسی آنها به وسیله پایشگر در مقابل الگوهای نقض نیازها استفاده می‌کنیم. این الگوهاکه در شکل (۵)نشان داده شده است، با یک همروندی نامطلوب تعیین می‌شود. این الگوهای همروند در شکل ترکیب رخدادها و ثابت‌ها روی ویژگی‌های Pr_1 تا Pr_3 ظاهر می‌شود (نک. بخش ۵-۱). برای مثال نقض $@T(cautionary)_\tau \wedge t(high)_{\tau.\tau+1}$ در شکل (۵)، یک نمونه همروند است که معرف رخداد ورود قطار به ناحیه هشدار در هنگام بالا بودن سرعت قطار (که یک ثابت است) است. برای شبیه‌سازی با استفاده از کتابخانه TPL، برای هر اجرای همروند، یک وظیفه در نظر می‌گیریم. سیستم کنترل قطار و محیط آن (قطار) با سه اجرای همروند مدل می‌شوند: (۱) رفتار قطار در ناحیه؛ یعنی ورود یا خروج قطار به یا از ناحیه، (۲) رفتار سرعت قطار یعنی افزایش یا کاهش سرعت و (۳) رفتار ترمز قطار یعنی گرفتن یا آزاد کردن آن. همچنین، یک وظیفه همروند برای اجرای برنامه پایشگر منظور می‌شود تا رخدادها و ثابت‌های گزارش شده به وسیله سه وظیفه فوق‌الذکر را دریافت کند.

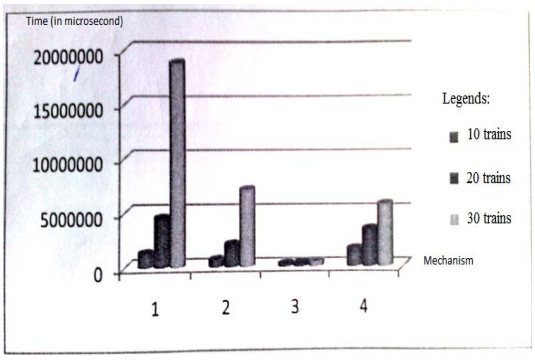
ما از کتابخانه TPL و نخ‌ها برای شبیه‌سازی اجراهای همروند استفاده می‌کنیم، اما کتابخانه TPL، دید روشن‌تری از رفتار سیستم و محیط آن می‌دهد. انتساب وظایف به هسته‌های پردازنده به صورت پویاست، در حالی در مکانیسم چندنخی انتساب هر نخ اجرا به یک هسته به صورت ایستاست. روش پویا با سوئیچ کردن بین هسته‌های یک پردازنده، یک اجرا را به هر هسته از پردازنده که بیکار است، می‌دهد در حالی که در روش ایستا، یک هسته وقف یک نخ می‌شود و از هسته بیکار پس از اتمام وظیفه‌اش استفاده نمی‌شود. در نتیجه، شبیه‌سازی شبکه‌های همروند با استفاده از کتابخانه TPL دید واقعی‌تری از همروندی در سیستم‌های حقیقی و محیط آنها را می‌دهد. علاوه بر این، وقتی تعداد فرآیندهای همروند افزایش می‌یابد، استفاده از کتابخانه TPL به کارآیی محاسباتی سطح بالاتری روی پردازنده‌های چند هسته‌ای نسبت به استفاده الگوی چندنخی منجر می‌شود.

شکل (۶) زمان اجرای شبیه سازی را با چهار مکانیسم نشان می‌دهد: (۱) تک نخی؛ (۲) چندنخی؛ (۳) مبتنی بر وظیفه و (۴) ترکیب چندنخی و مبتنی بر وظیفه. ما این مکانیسم‌ها را روی یک پردازنده دو هسته برای ۱۰، ۲۰ و ۳۰ قطار به کار بردیم .شکل (۶) زمان اجرا را نشان می‌دهد که در آن محور عمودی نشان دهنده زمان و محور افقی نشان دهنده مکانیسم‌های ۱ تا ۴ است. میله‌ها زمان اجرا را برای ۱۰، ۲۰ و ۳۰ قطار نشان می‌دهد. همان طور که شکل

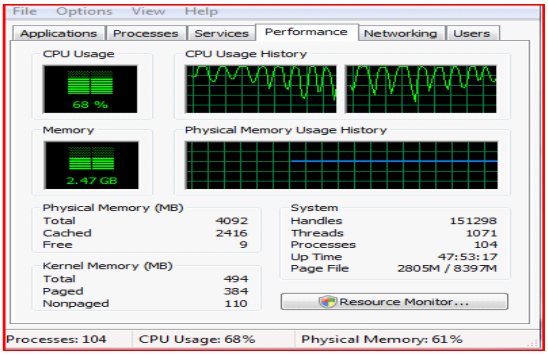
(۶) نشان می‌دهد، مکانیسم مبتنی بر وظیفه حداقل زمان اجرا را دارد. شکل‌های (۷) تا (۹)بهره‌وری هسته‌ها را در استفاده از مکانیسم‌ها برای ۵۰ قطار نشان می‌دهد. در حالی که شکل‌های (۷) و (۹) نشان می‌دهد که مکانیسم‌های چندنخی و ترکیبی (چندنخی و مبتنی بر وظیفه) از ۶۸٪ و ۵۴٪ و هسته‌ها استفاده کرده‌اند، شکل (۸) نشان می‌دهد که مکانیسم مبتنی بر وظیفه از هسته‌ها نزدیک ۱۰۰٪ بهره می‌گیرد.

۶- ارزیابی کارآیی روش پیشنهادی

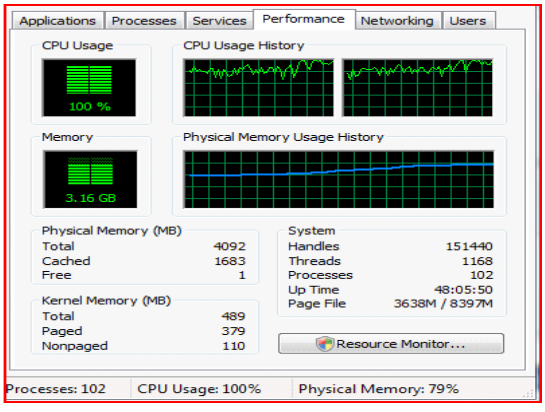
این بخش کارآیی روش پیشنهادی را ارزیابی می‌کند. ما روش پیشنهادی را برای نیازهای ایمنی کاربران سیستم حفاظت قطار به کار گرفتیم تا پایشگر سیستم را مدل و شبیه‌سازی کنیم. ارزیابی قدم‌های روش پیشنهادی به شرح زیر است: (۱) کارآیی مستندسازی نیازهای کاربران. در ارتباط با سیستم‌های حساس به ایمنی، کارشناس سیستم مستندات نیازها را مرور می‌کند تا مطمئن شود که آنها سازگار و صحیح هستند. چنین مرورهایی کارآ خواهد بود اگر مستندات واضح و قابل فهم برای کارشناس باشد. برای محقق شدن این ویژگی، ما نیازهای ایمنی مبتنی بر دامنه مسأله را به زبان کارشناس و خبره سیستم بیان کردیم و سپس انتزاع کمیت‌های سیستم را با تقسیم آنها به کمیت‌های پایشی و کنترلی نشان دادیم. (۲) کارآیی توصیف نیازها. در سیستم‌های حساس به ایمنی، توصیف نیازها باید قابل وارسی به صورت رسمی (مبتنی بر ریاضی) باشد تا بتوان درستی و سازگاری آنها را نشان داد. (۳) کارایی طراحی نیازها. با نگاشت نیازهای ایمنی به یک اتوماتا، یک مدل انتزاعی ساختیم که یک طراحی کارآ را برای پیاده‌سازی پایشگر فراهم‌کرد. برای نشان دادن کارایی طراحی، آن را برای پایشگر کنترل سرپرستی حفاظت قطار به کار بردیم.



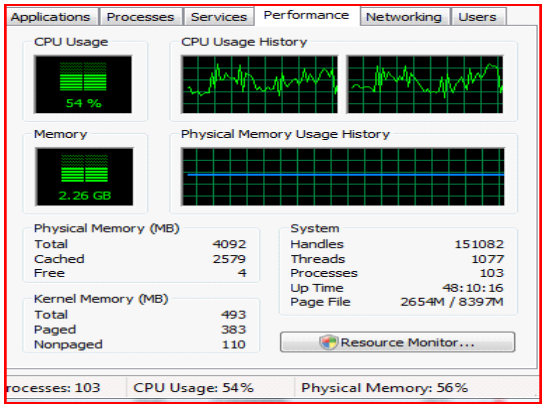
شکل (۶): زمان اجرا با مکانیسم‌های (۱) تک نخی، (۲)چند نخی، (۳) وظیفه‌ای و (۴) ترکیبی



شکل (۷): بهره‌برداری از هسته‌ها در برنامه چند نخی برای ۵۰ قطار



شکل (۸): بهره‌برداری از هسته‌ها در برنامه چند وظیفه‌ای برای ۵۰ قطار



شکل (۹): بهره‌برداری از هسته‌ها در برنامه ترکیب چند وظیفه‌ای و چند نخی

۷- کارهای مرتبط

در مقایسه با تحقیق‌های دیگران، دو مرحله را در نظر می‌گیریم: (۱) توصیف مبتنی بر مدل و (۲) شبیه‌سازی پایشگر، زیرا روش ما شامل دو بخش بود: (۱) توصیف مبتنی بر مدل شامل توصیف سطح بالا (مبتنی بر رخداد) و توصیف مبتنی بر حالت و (۲) شبیه‌سازی پایشگر در کنترل سرستی سیستم های رخداد گسسته. بنابراین، ما تحقیق‌های مرتبط به توصیف و شبیه سازی سرپرستی را در نظر می‌گیریم. برادین و ونام چارچوبی ارائه کردند که در آن از رخدادهای زمان‌دار برای توصیف کنترل سرپرستی سیستم‌های گسسته استفاده شدند[۱۴]. آنها تنها به موارد نظری در طراحی کنترل سرپرستی پرداختند. جیانگ و کومار [۱۵] و رادیا و سارگنت [۱۶] از منطق برای توصیف استفاده کردند که جیانگ و کومار از منطق زمانی انشعابی CTL و رادیا و سارگنت مشابه با روش ما، رخداد و شرط (ثابت) را برای توصیف استفاده کردند. اما آنها عموماً تنها یک توصیف مبتنی بر رخداد ارائه کردند و روشی برای نگاشت توصیف‌های مبتنی بر رخداد به توصیف‌های مبتنی بر حالت پیشنهاد ندادند. در حالی که برخی از محققان توصیف‌های مبتنی بر رخداد و منطق را مشخص کرده‌اند [۱۴-۱۶] ، برخی دیگر به توصیف‌ها مبتنی بر حالت پرداختند. در این میان، شبکه های پتری عمومی‌ترین اتوماتا برای توصیف‌ها هستند [۱۷-۲۰] . ما روی رفتار پایشگر و شبیه‌سازی و پیاده‌سازی آن تأکید کردیم. در این رابطه، گروه لگراند برای شبیه‌سازی از شی‌های نخی همروند و زبان جاوا استفاده کردند [۲۱]. دو تشابه بین روش لگراند و متد ما وجود دارد: (۱) استفاده از مدل‌سازی داده‌ها در توصیف و (۲) همروندی و مکانیسم چندنخی در شبیه‌سازی. اما گروه لگراند از مکانیسم چندنخی در شبیه‌سازی همروندی استفاده کردند و برای استفاده از مکانیسم ترکیب وظیفه و چندنخی و مکانیسم مبتنی بر وظیفه تلاشی نکردند. شبیه‌سازی ما نشان داد که مکانیسم مبتنی بر وظیفه نسبت به دو مکانیسم دیگر کارآتر است. علاوه بر این، در روش ما، قیودی که وسیله شبیه‌سازی استفاده می‌شوند، با استفاده از ثابت‌ها در بخش توصیف مشخص شدند. سرپرستی براساس ثابت‌ها، روش مؤثری در کنترل سرپرستی با استفاده از شبکه‌های پتری است. به این دلیل است که آیورداک و آنت‌ساکلیس به اهمیت استفاده از ثابت‌ها در سرپرستی پرداخته‌اند [۲۲].

همان طور که در بخش مقدمه بیان شد، هدف این مقاله مشخص‌کردن مسأله مشاهده و پایش رفتار سطح پایین حین اجرای نرم‌افزار و وارسی درستی آن در برابر توصیف سطح بالای نیازهای کاربران سیستم است. این مسأله تاکنون توجه محققان دیگر را به خود جلب کرده است. گادین و باگناتو رویکردی را با استفاده از نظریه کنترل برای تثبیت نگاهداری نرم‌افزار ارائه‌کرده‌اند [۵]. این رویکرد، نرم افزار مورد وارسی را قبل از استفاده تغییر می‌دهد تا بتواند پایش شود و با سرپرست در زمان اجرا تعامل کند (یعنی گزارش حالت‌های خود را به سرپرست گزارش کند). رفتار امن با یک ماشین حالت متناهی توصیف می‌شود. مشابه با روش ما، آنها نخست یک توصیف رسمی هم از نرم‌افزار مورد وارسی و هم از نیازهای کاربر ارائه کردند و سپس به جای اثبات ارضای نیازها به وسیله نرم‌افزار، رفتار نرم‌افزار را در زمان اجرا کنترل کردند تا نیازها را اعمال کنند. اما این روش، توصیف رسمی خود را با یک روش مبتنی بر حالت (اتوماتا) آغاز کرد که فاقد : (۱) استفاده از داده‌های دامنه مسأله که بر حسب واژگان کارشناس سیستم بیان می‌شود و (۲) توصیف مبتنی بر رخداد است. ما در این مقاله با گره زدن توصیف‌ها در سطح کاربر و توصیف‌های مبتنی بر رخداد به توصیف‌های مبتنی بر حالت، راه حلّی برای چالش نگاشت توصیف سطح بالا به توصیف سطح پایین ارائه دادیم.

این مقاله و همچنین گادین و باگناتو [۵] و گروه ونگ [۲۳] از برنامه‌های پایشگری که از نظریه کنترل استفاده می‌کنند و پایش حین اجرا را از طریق کنترل سرپرستی به‌کار می‌گیرند، استفاده کردند. گروه ونگ مسأله اعتبارسنجی کنترل پویای اجرای نرم‌افزار همروند را به منظور اجتناب از بن‌بست‌ها مطرح کردند. آنها، استفاده از شبکه‌های پتری را در مدل‌سازی ارائه و قیود را با استفاده از ثابت‌ها بیان

کردند. بنابراین، آنها تنها از توصیف مبتنی بر حالت استفاده کردند و به توصیف‌های در سطح کاربر و توصیف‌های مبتنی بر رخداد توجهی نکردند. گروه کالان از نظریه کنترل سرپرستی و اتوماتای متناهی برای پایش نرم‌افزارهایی که باید دارای اطمینان بالا باشند، استفاده کردند [۲۴] و گروه برینگر به رابطه بین کنترل سرپرستی و پایش و وارسی حین اجرا پرداختند [۲۵] .

۸- نتایج و کارهای آینده

در این مقاله، یک روش سه مرحله‌ای برای مدل‌سازی و شبیه‌سازی پایشگر در کنترل سرپرستی سیستم‌های گسسته ارائه شد. این روش با ویژگی‌های محیطی که به وسیله کاربر سیستم توصیف می‌شود، شروع شد و سپس با توصیف‌های مبتنی بر رخداد و مبتنی بر حالت برای نیازهای ایمنی ادامه یافت. این توصیف‌ها باید در نهایت، به وسیله سیستم مورد سرپرستی رعایت شود. در نهایت این مقاله با شبیه‌سازی پایشگر با استفاده از کتابخانه TPL خاتمه یافت. روش ارائه شده یک روش افزایشی بود که توصیف نیازهای ایمنی انتزاعی و سطح بالا را به فعالیت‌های نرم‌افزاری سطح پایین و حین اجرا نگاشت‌کرد. برای نشان دادن به‌کارگیری روش ارائه شده در این مقاله، سه قدم آن را برای سیستم حفاظت قطار به‌کارگرفتیم.

به منظور سازماندهی بحث گسترده فوق‌الذکر،آن را براساس سه محور مقوله‌بندی می‌کنیم: (۱) بهره‌گیری از دو نوع توصیف برای نمایش دو دید از رفتار سیستم، یکی برای نمایش تعامل‌های سیستم با محیطش برحسب رخدادها و دیگری برای نمایش رفتار مبتنی بر حالت سیستم. با گره زدن این دو نوع توصیف، ما راه حلّی برای چالش نگاشت توصیف‌های سطح بالا به توصیف‌های سطح پایین ارائه دادیم. (۲) ارائه متد سازنده‌ای برای استفاده از سه سطح سلسله مراتبی از توصیف‌های: (الف) سطح کاربر؛ (ب) سطح توصیف و (ج) سطح پیاده‌سازی. توصیف انتزاعی نیازها و گسترش استفاده از متد پیشنهادی، دو فایده جنبی سطح سلسله مراتبی توصیف‌هاست. (۳) استفاده از الگوی طراحی مبتنی بر حالت برای تسهیل پیاده‌سازی رفتار پویای سیستم با استفاده از اتوماتای پتری نت. به منظور بررسی کارآیی، ما زمان اجرای شبیه‌سازی‌هایی را که از مکانیسم‌های چندنخی و مبتنی بر وظیفه استفاده کرد، مقایسه کردیم. بررسی‌های عملی نشان داد که مکانیسم مبتنی بر وظیفه محض، کاراترین روش برای پیاده‌سازی همروندی وظایف در سیستم‌های رخداد گسسته است. علاوه براین، انتساب پویای وظایف به هسته‌های پردازنده که به وسیله کتابخانه TPL فراهم می‌شود، زمان اجرا را نسبت به انتساب ایستای وظایف به هسته‌ها کاهش داد.

به عنوان یک تحقیق آتی، کار ارائه شده در این مقاله می‌تواند برای سیستم‌های SCADA[۱۲] گسترش یابد. نمونه‌ای از این سیستم‌ها، سیستم‌های رایانه‌ای هستند که فرایندهای صنعتی را پایش و کنترل می‌کنند [۲۶]. سیستم‌های SCADA معمولاً شامل یک پایگاه داده توزیع شده است که در آن داده‌ها، مقادیری هستند که به وسیله سیستم پایش و کنترل می‌شوند و همراه با زمان محاسبه‌شان ذخیره می‌شوند. در این پایگاه داده، سابقه داده‌ها یک دنباله از جفت‌های زمان – مقدار را تشکیل می‌دهد. برای به کارگیری روش پیشنهادی این مقاله برای یک سیستم SCADA، باید یک روش خاص برای توصیف ویژگی‌های محیط سیستم و همچنین، تقسیم بازه‌ها درنظر گرفته شود، زیرا داده‌ها زمانی و توزیع شده هستند.

این تحقیق توسط دانشگاه کاشان بر اساس قرارداد طرح پژوهشی شماره ۳۰۵۵ پشتیبانی شده است.

مراجع:

[1] P. Ramadge and W. Wonham. The control of discrete event systems, Proceedings of IEEE, Special Issue on Discrete Event Dynamic Systems, pp. 81–98, 1989.
[2] D. Leijen, W. Schulte, and S. Burkhardt. The Design of a Task Parallel Library, ACM SIGPLAN Notice, 4 (10), pp. 227-242, 2009
[3] J. Sharp. Microsoft Visual C# 2010 Step by Step, Microsoft Press, 2010.
[4] S. Lipschutz. Schaum's outline of theory and problems of set theory and related topics. McGraw-Hill, 1998.

[5] B Gaudin and A. Bagnato. Software Maintenance through Supervisory Control, the 34th IEEE Software Engineering Workshop (SEW), pp. 97-105, 2011.

[6] Petri Nets Fundamental Models, Verification and Applications, In M. Diaz, (Ed.), Wiley, 2009.

[7] X. D. Koutsoukos and P. J. Antsaklis. Hybrid Control Systems Using Timed Petri Nets: Supervisory Control Design Based on Invariant Properties, Lecture Notes in Computer Science, Vol. 1567/1999, 67, Springer, 1999.

[8] M. Iordache and P.J. Antsaklis. Supervisory Control of Concurrent Systems: A Petri Net Structural Approach, Springer, 2006.

[9] G.J. Tsinarakis and N.C. Tsourveloudis. Adding two level supervisory control in the Hybrid Petri Net methodology for Production Systems, in Proceedings of the 17th Mediterranean Conference on Control and Automation, pp. 1090 – 1095, 2009.

[10] Z. Chun-Fu and L. Zhi-Wu. Design of Liveness-Enforcing Supervisors via Transforming Plant Petri-Net Models of FMS, Journal of Asian Control, Special Issue on Petri-Nets for System Control and Automation, Vol.12, No 3, pp. 240–252, 2010.

[11] L. Feihua , W. Weimin, S. Hongye and C. Jian. Non-blocking Decentralized Control of Discrete Event Systems Based on Petri-Nets, Journal of Asian Control, Special Issue on Petri-Nets for System Control and Automation, Vol.12, No.3, pp. 323–335, 2010,

[12] M.V. Iordache and P.J. Antsaklis. Concurrent program synthesis based on supervisory control, In Proceedings of American Control Conference, pp. 3378 – 3383, 2010.

[13] F. Basile and P. Chiacchio. On the Implementation of Supervised Control of Discrete Event Systems, IEEE Transactions on Control Systems Technology, Vol.15, No.4, pp. 725-739, 2007.

[14] B.A. Brandin and W.M. Wonham. Supervisory Control of Timed Discrete Event Systems, IEEE Transactions on Automatic Control, Vol.39, No.2, pp. 329- 342, 1994.

[15] S. Jiang and R. Kumar. Supervisory Control of Discrete Event Discrete Systems with CTL* Temporal Logic Specifications, SIAM J. Control Optim., Vol.44, No.6, pp. 2079-2103, 2006.

[16] A. Radiya and R.G. Sargent. A Logic-Based Foundation of Discrete Event Modeling and Simulation, ACM transactions on Modeling and Computer Simulation, Vol.4, No.1, pp. 3-51, 1994.

[17] M. V. Iordache and P. J. Antsaklis. Concurrent Program Synthesis Based on Supervisory Control, American Control Conference, pp. 3378-3383, 2010.

[18] Y. Song, J. Kim and J. Lee. Modeling User Specification Based on Supervisor Control and Petri nets, The International Conference on Computational Intelligence for Modeling Control & Automation, IEEE Society, 76-81, 2008.

[19] J. Flochova. A Petri net based supervisory control implementation, The IEEE International Conference on Systems, Man and Cybernetics, Vol.2, pp. 1039-1044, 2003.

[20] C. Bobeanu, E. J. H. Kerckhoffs and H. V. Landeghem, Modeling of Discrete Event Systems: A Holistic and Incremental Approach Using Petri Nets, ACM Transactions on Modeling and Computer Simulation, Vol.14, No.4, pp. 389-423, 2004.

[21] I. Legrand. Multi-threaded, discrete event simulation of distributed computing systems, Computer Physics Communications, Elsevier, vol. 140, pp. 274-285, 2001.

[22] M. V. Iordache and P. J. Antsaklis. Supervision Based on Place Invariants: A Survey, Journal of Discrete Event Dynamic Systems, Springer, Vol.16, No.4, pp. 451-492, 2006.

[23] Y. Wang, T. Kelly, M. Kudlur, S. Mahlke and S. Lafortune. The Application of Supervisory Control to Deadlock Avoidance in Concurrent Software, The 9th international Workshop on Discrete Event Systems, pp. 287-292, 2008.

[24] S. Callanan et al. Software Monitoring with Bounded Overhead, The IEEE International Symposium on Parallel and Distributed Processing, pp. 1-8, 2008.

[25] H. Barringer, D. Gabbay and D. Rydeheard. From Runtime Verification to Evolvable Systems, The 7th International Conference on Runtime verification, pp. 97-110, 2007.

[26] S. A. Boyer. "SCADA: Supervisory Control and Data Acquisition", ISA: The Instrumentation, Systems, and Automation Society, 4th Ed, 2009.

زیرنویس‌ها:

[1] Discrete Event System
[2] Supervisor
[3] Supervisory Control Theory
[4] Invariants
[5] Safety Requirements
[6] Asynchronous
[7] Colored Petri-Net
[8] Marking
[9] Marks
[10] Enabled
[11] Fire
[12] Supervisory Control And Data Acquisition