تولید قوانین راستی‌آزمایی از توصیف‌های اتوماتای زمانی

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

مهدی برهانی دهکردی
دانشجوی کارشناسی ارشد، مهندسی کامپیوتر
mahdi.borhanidehchordi@hotmail.com

مصطفی امینی
دانشجوی کارشناسی ارشد، مهندسی کامپیوتر
mostafa.amini@yahoo.co

چکیده

تولید قوانین راستی‌آزمایی نرم‌افرار از روی توصیف مسئله، از مسائل مطرح در مهندسی نرم‌افزار است. از آنجا که آتاماتای‌زماندار یکی از روش‌های رسمی و توانمند در توصیف یک مسئله با قیود زمانی است، ما در این مقاله به ارائه روشی می‌پردازیم که به وسیله آن می‌توان از توصیف‌های اتاماتای‌زماندار، قوانین راستی‌آزمایی را تولید کرد. این قوانین برای راستی‌آزمایی رفتار نرم‌افزاری که از روی توصیف مسئله ساخته می‌شود، استفاده می‌شود تا رفتار مخاطره‌آمیز نرم‌افزار را اعلام کند. روش ما سه مرحله دارد: (۱) برای هر مولفه آتاماتای‌زماندار که مجموعه‌ای از حالات و گذارهای بامعنای اتوماتا را مشخص می‌کند، یک گزاره بر حسب حساب‌رخداد تولید می‌کنیم، (۲) گراف دسترسی حالات آتامای‌زماندار را به دست می‌آوریم که در آن ترکیب مخاطره‌آمیز حالات مشخص شده است، (۳) برای هر ترکیب مخاطره‌آمیز حالات، یک گزاره بر حسب حساب‌رخداد با توجه به گزاره‌های بند ۱ تعیین می‌کنیم. این گزاره‌ها، قوانین راستی‌آزمایی را تشکیل می‌دهند. در خاتمه با طرح یک مسئله، استفاده از روش فوق‌الذکر را در تولید قوانین راستی‌آزمایی نشان می‌دهیم.

واژه‌های کلیدی: آتاماتای زمانی، راستی‌آزمایی، حساب رخداد

۱- مقدمه

راستی‌آزمایی نرم‌افزار یکی از دغدغه‌های مهم در مهندسی نرم‌افزار است که ارائه یک روش مناسب برای آن همواره مورد توجه بوده است. یک روش برای راستی‌آزمایی نرم‌افزار این است که از توصیف مسئله، قوانینی بسازیم که معرف رفتار مخاطره آمیز نرم‌افزار باشد و سپس رفتار نرم‌افزار را در برابر این قوانین راستی‌آزمایی کنیم. یک رفتار مخاطره‌آمیز، مجموعه‌ای از حالات نرم‌افزار و محیط آن است که منجر به ایجاد خطر می‌شود. برای مثال، در یک سیستم کنترل تقاطع جاده و ریل آهن، یک مانع با گرفتن فرمان از سیستم، جاده را در برابر قطار (محیط) کنترل می‌کند. در این سیستم، مجموعه حالات "مانع بالا است" و "قطار در تقاطع قرار دارد" یک رفتار مخاطره‌آمیز است. برای تولید قوانین راستی‌آزمایی یک توصیف مناسب از مسئله ضروری است. در این راستا، آتاماتای‌زماندار [۱]، یک روش دیداری مناسب برای توصیف سیستم‌های زمان‌واقعی[1] و رخدادگرا است و توصیف یک مسئله به وسیله آنها، فهم مسئله را آسان می‌سازد.

ما در این مقاله به ارائه روشی می‌پردازیم که به وسیله آن می‌توان از توصیف یک مسئله که به وسیله آتاماتای‌زماندار انجام شده است، قوانین راستی‌آزمایی را تولید کنیم. دلیل اینکه آتاماتای زماندار یک روش دیداری است،

برای تولید قوانین راستی‌آزمایی از یک آتاماتای‌زماندار، ما یک گراف دسترسی[2] از ترکیبی از حالات مختلف آتاماتا می‌سازیم و سپس با استفاده از یک منطق زمانی به تولید قوانین راستی‌آزمایی می‌پردازیم.

ما نشان می‌دهیم که هنگامی که ویژگی‌های آتاماتای زماندار با ویژگی‌های منطق زمانی حساب رخداد[۲] آمیخته می‌شود، تولید قوانین راستی‌آزمایی میسر می‌شود.

در ادامه، در بخش دوم، به شرح آتاماتای زماندار و در بخش سوم به شرح منطق زمانی حساب رخداد می‌پردازیم. در بخش چهارم، ابتدا مرحله اول رویکرد پیشنهادی خود را که شامل استخراج گزاره‌های حساب رخداد از ساختارهای آتاماتای زمانی است، شرح می‌دهیم و سپس به مرحله دوم و مرحله سوم رویکرد پیشنهادی خود که به ترتیب شامل ساخت گراف دسترسی و تولید قوانین راستی‌آزمایی است، می‌پردازیم. این مرحله را با طرح یک مسئله نشان می‌دهیم. در نهایت در بخش پنجم، نتیجه‌گیری خود را از این تحقیق بیان می‌کنیم.

۲- آتاماتای‌زماندار

برای کنترل رفتار یک سیستم زمان‌واقعی به یک مدل محاسباتی که دارای مفهوم زمان باشد نیاز است .برای این منظور، آتاماتای‌زماندار یک روش ساده و توانمند برای شرح و تفسیر نمودارهای انتقال- حالت با قیود زمانی می باشد.

آتاماتای زماندار [۱] توسط آقای Alur مطرح و به کمک آقای Dill تکمیل شد. آتاماتای‌زماندار یک ماشین حالت متناهی است که به مجموعه‌ای از ساعت‌ها مجهز شده است. ساعت‌ها، توابع حقیقی با مقدار پیوسته‌ای از زمان هستند که بصورت مجزا زمان‌های بین رخدادها را ثبت می‌کنند. همه ساعت‌ها بصورت همگام افزایش می‌یابند. آتاماتای‌زماندار بعنوان یک نمادگذاری رسمی برای مدلسازی رفتار سیستم‌های زمان‌واقعی معرفی شد. آتاماتای زماندار یک روش کلی برای نمایش نمودارهای انتقال- حالت با محدودیت‌های زمانی است که از تعدادی از متغیرهای زمانی که مقدار حقیقی دارند، استفاده می‌کند.

[1] Real Time

[2] Reachability Graph

آتاماتای زماندار برای وارسی سیستم‌های توزیع‌شده، بهینه‌سازی، وارسی برنامه‌های چندوظیفه‌ای، تحلیل شبکه، برنامه‌ریزی و زمانبندی نیز مناسب است [۳ و ۴ و ۵ و ۶]. در مفهوم زمانبندی، هدف پیدا کردن یک مسیر از حالت اولیه به یک حالت نهایی است و این درحالی است که تمام عمل‌ها انجام شده و تمام نیازهای زمانی نیز ارضاء شوند . اگر چنین مسیری یافت شود که یک استاندارد بهینه‌سازی را برآورده کند ، راه حل مسئله زمانبندی بدست آمده است .

یک آتاماتای‌زماندار در واقع ، یک چندتایی $M=(\sum,S,S0,X,E)$ می‌باشد که در آن $\sum$ یک مجموعه متناهی از عمل‌ها (Action)، S مجموعه متناهی از حالت‌ها ، $S0$ مجموعه متناهی از حالات اولیه ($S0 \subset S$)، و X مجموعه متناهی از ساعت‌ها و E مجموعه‌ای از انتقال‌ها، که هر انتقال تشکیل شده است از : $< L , a , g , \lambda, L' >$. انتقال از حالت فعلی L ، به حالت بعدی L' وقتی که عمل a انجام شود و g (X) محدودیت ساعت با مقدار درست(True) ارزیابی شوند صورت می‌گیرد وλ زیر مجموعه‌ای از X است که با این انتقال بازنشانی می‌شود.

$$g::=x\le c \mid c\le x \mid x-y\le c \mid x<c \mid c<x \mid x-y<c \mid g\wedge g \mid true$$

که $x,y \in X , c \in R$.

۳- منطق زمانی حساب رخداد

منطق زمانی حساب‌رخداد [۲] به وسیله Kowalski و همکارش بر اساس منطق رتبه اول[3] برای نمایش رخدادها بر حسب دوره‌های زمانی معرفی شد. این منطق شامل نوع‌های "رخداد یا کنش"، "روان‌گزاره" و "زمان" است. روان‌گزاره، یک کمیت مانند "درجه حرارت" یا یک گزاره مانند "دستگاه خراب است"، می‌باشد که مقدار یا درستی آن در طول زمان تغییر می‌کند. دستگاه استنتاجی حساب رخداد: (۱)کنش‌ها را بر حسب زمان و (۲) تاثیرات کنش‌ها را به عنوان ورودی می‌گیرد و (۳) حقایق را در خروجی تولید می‌کند. ما در رویکرد پیشنهادی از نوع ساده‌شده حساب‌رخداد به نام SEC[4] [۷] که نقاط زمانی را به جای دوره‌های زمانی به کار می‌برد، استفاده می‌کنیم. SEC، شامل یک اصل مرکزی S است (فرمول شماره ۱) که دارای سه گزاره Happens، Initiates و Terminates در فرض و یک گزاره HoldsAt در نتیجه است. اصل S بیان می‌دارد که روان‌گزاره β به واسطه رخداد α_1 در گذشته برقرار شده است (true) و تا لحظه جاری، رخدادی رخ نداده است که آن را نادرست (false) ساخته باشد. گزاره‌های SEC و شرح آنها در جدول شماره ۱ نشان داده شده است. تاثیرات کنش‌ها با Initiates و Terminates و حقایق با HoldsAt مشخص می‌شوند. α معرف رخداد یا عمل، β معرف روان‌گزاره و τ معرف نقطه‌ای از زمان است.

S Axiom:

$$HoldsAt(\beta , \tau) \leftarrow Happens(\alpha_1 , \tau_1) \wedge Initiates(\alpha_1 , \beta) \wedge \tau_1 < \tau \wedge \sim Clipped(\tau_1 , \beta ,\tau)$$

$$\sim Clipped(\tau_1 , \beta ,\tau) \equiv \sim (\alpha_2 , \tau_2): Happens(\alpha_2 , \tau_2) \wedge Terminates(\alpha_2 , \beta) \wedge \tau_1 < \tau_2 \wedge \tau_2 < \tau$$

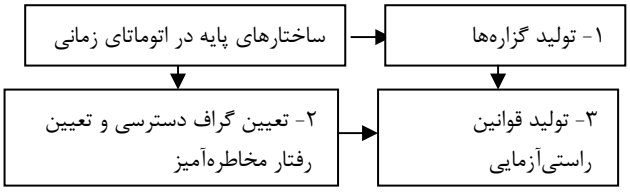
فرمول شماره ۱- اصل مرکزی S **در** SEC [۷]

جدول شماره ۱- گزاره‌های SEC [۷]

گزاره	معنی
$Initiates(\alpha,\beta)$	گزاره β پس از رخداد α برقرار است
$Terminates(\alpha,\beta)$	گزاره β پس از رخداد α برقرار نیست
$Initially_P(\beta)$	گزاره β از ابتدا برقرار بوده است
$Happens(\alpha,\tau)$	رخداد α در زمان τ رخ می‌دهد
$HoldsAt(\beta,\tau)$	گزاره β در زمان τ برقرار است
$Clipped(\tau_1,\beta,\tau_2)$	گزاره β بین زمان τ_1 و زمان τ_2 خاتمه می‌یابد
$\tau_1 < \tau_2$	زمان τ_1 قبل از زمان τ_2 قرار دارد

۴- رویکرد پیشنهادی

ما در این بخش به ارائه روشی برای تولید قوانین راستی‌آزمایی از توصیف‌های اتوماتای‌زماندار می‌پردازیم. این قوانین برای راستی‌آزمایی رفتار نرم‌افزاری که از روی توصیف مسئله ساخته می‌شود، استفاده می‌شود. روش ما سه مرحله دارد (شکل شماره ۱). در مرحله اول، هدف این است که یک دستگاه از گزاره‌ها از توصیف اتوماتای یک مسئله بر حسب حساب‌رخداد ارائه دهیم. ما از این گزاره‌ها در مرحله سوم برای تولید قوانین راستی‌آزمایی بهره می‌گیریم. برای به دست‌آوردن دستگاه گزاره‌ها، ابتدا ساختارهای پایه در آتاماتای‌زماندار را بیان و سپس گزاره‌های متناظر با هر ساختار را بر حسب حساب‌رخداد به دست می‌آوریم. از آنجا که یک آتاماتای‌زماندار مجموعه‌ای بامعنا از ساختارهای پایه زمانی است، دستگاه گزاره‌های حاصل، یک توصیف رسمی و بامعنا از آتاماتا بر حسب حساب‌رخداد خواهد بود. در مرحله دوم، هدف این است که رفتار مخاطره آمیز را در مجموعه رفتارهای ممکن اتاماتا به دست آوریم. برای نمایش رفتارهای ممکن آتاماتا گراف دسترسی آن را به دست می‌اوریم تا در آن رفتار مخاطره‌آمیز را مشخص کنیم.



شکل شماره ۱ – روش پیشنهادی برای تولید قوانین راستی‌آزمایی

[3] First Order Logic
[4] Simplified Event Calculus

گراف دسترسی یک آتاماتای زمانـدار، یـک سـاختار سلـسله مراتبـی از ترکیب‌های بامعنا از حـالات مختلـف آتاماتـا اسـت کـه در آن برخـی از ترکیب‌ها، معرف رفتار مخاطره‌آمیز ممکن است. در مرحله سـوم، هـدف تولید قوانین راستی‌آزمایی است.از آنجا که هر ترکیـب مخـاطره‌آمیـز در گراف دسترسی، یک ترکیب با معنا از ساختارهای پایه زمانی بـه دسـت آمده در مرحله اول است، گزاره‌های متناظر با این ترکیب‌ها را بـا توجـه به گزاره‌های مرحله اول تعیین می‌کنیم. این گزاره‌هـا، قـوانین راسـتی-آزمایی را تشکیل می‌دهند. ما در بخش ۴-۲، ایـن روش را بـرای تولیـد قوانین راستی‌آزمایی سیستم تقاطع ریل و جاده به کار می‌بریم.

۴-۱- تولید گزاره‌های منطقی از ساختارهای پایه آتاماتای‌زماندار

ساختار ۱: تغییر حالت. درسیستم‌های زمان‌واقعی تغییر حالت در اثر رخ دادن یک رخداد اتفاق می‌افتد.

تعریف بصری: در شکل شماره۲ زمانی که در حالت L0 قرار داشته باشیم و رخداد e اتفاق بیافتد از حالت L0 خارج شده و وارد حالت L1 می شویم.

تولید گزاره برای ساختار ۱: زمانیکه رخداد e در زمان T اتفاق می-افتد حالت L0 غیر فعال (false) می شود و زمانیکه رخداد e در زمانT اتفاق می افتد حالت L1 در زمان 'T فعال (true) می شود که این دستورات معادل تغییر حالت در آتاماتای‌زماندار است.

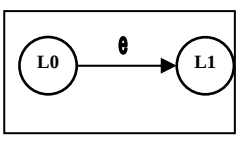
$$HoldsAt(L1,T') \longleftarrow Happens(e,T),\ Terminates(e,L0,T),\ Initiates(L1,T') \quad (T<T')$$

گزاره شماره ۱ برای ساختار شکل شماره ۲

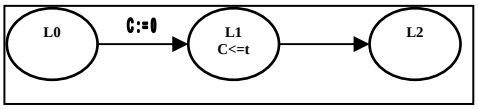
ساختار ۲: قرار داشتن در یک حالت. در سیستم های زمان‌واقعی با قرار داشتن سیستم در یک حالت وضعیت سیستم درآن لحظه تعیین می‌شود.

تعریف بصری: با توجه به شکل شماره ۳ زمانی که سیـستم در حالـت L0 قرار داشته باشد این حالت مشخص کننده وضـعیت سیـستم در آن لحظه خاص است.

تولید گزاره برای ساختار ۲: قرار گرفتن سیستم را در یک حالت را به صورت HoldsAt (L_0,T) نشان می‌دهیم. این گزاره معرف این است که حالت L0 در زمانT فعال (true) است (بعبارت دیگر سیستم در زمان T در حالت L0 قرار دارد). برای نشان دادن قرار نداشتن در حالت L0 در زمان T از گزاره ¬HoldsAt (L_0,T) استفاده می‌کنیم.



شکل شماره ۲- تغییر حالت در آتاماتای زماندار



شکل شماره ۳ – برقراری حالت شرطی (L1, C<=t) در آتاماتای زمانی C=Clocck

$$HoldsAt(L0,T') \longleftarrow HoldsAt(L0,T) \quad (T<T')$$
یا
$$HoldsAt(L1,T') \longleftarrow Happens(e,T),\ Terminates(e,L0,T),\ Initiates(L1,T') \quad (T<T')$$

گزاره شماره ۲ برای ساختار ۲

ساختار ۳: توقف و تغییر حالت. در برخی حالات سیستم باید مدتی در یک حالت باقی بماند و سپس تغییر حالت دهد.

تعریف بصری: زمانیکه سیستم در حالت L1 قرار دارد باید حداقل t واحد زمانی در این حالت باقی بماند و سپس موقعیکه رخداد گذشت زمان اتفاق افتاد و شرط برقرار شد از حالت L1 وارد حالت L2 شود (شکل شماره ۳).

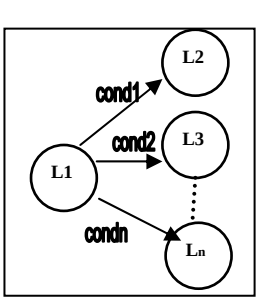
تولید گزاره برای ساختار ۳: اگر در حالت L1 قرار داشته باشیم و در زمان T1 یک رخداد رخ دهد (در اینجا رخداد ما گذشت زمان می باشد) که حالت L1 را غیر فعال کند در T1+T2 واحد زمانی بعد سیستم وارد حالت L2 می شود (بعبارت دیگر حالت L2 فعال می شود).

$$AntiTrajectory(L_1,T_1,L_2,T_2)$$

گزاره شماره ۳ برای ساختار شکل شماره ۳

ساختار ۴: تغییر حالت در صـورت برقـراری شـرط. در برخـی حـالات تغییر حالت در صورت برقراری شرط صورت مـی‌گیـرد و تنهـا رخ دادن رویداد کفایت نمی‌کند.

تعریف بصری: در آتاماتای زماندار ممکن است برخی از تغییر حالت‌ها وابسته به شرط روی گذار باشد و اگر شرط برقرار باشد تغییر حالت هم صورت می‌گیرد و در صورت برقرار نبودن شرط‌های روی گذارها هیچ تغییر حالتی را نخواهیم داشت این نوع تغییر حالت در شکل ۴ نشان داده شده است. در شکل شماره۴ شرط‌های روی گذارها باید منحصربفرد باشند و در هر لحظه باید یکی از این شرط‌ها بر قرار باشد



شکل شماره ۴ - تغییر حالت شرطی در آتامای زمانی

تولید گزاره برای ساختار ۴: در صورتی در زمان T، حالت Li (i=2..n) فعال (true) می‌باشد که در این زمان رخدادی رخ داده باشد که باعث غیر فعال شدن حالت L1 شده باشد و همچنین شرط روی گذار هم برقرار باشد.

$HoldsAt(L_2,T) \leftarrow Happens(e,T), Terminates(e,L_1,T), Holds\text{-}at(cond_1,T)$
$HoldsAt(L_3,T) \leftarrow Happens(e,T), Terminates(e,L_1,T), Holds\text{-}at(cond_2,T)$
.
.
.
$HoldsAt(L_n,T) \leftarrow Happens(e,T), Terminates(e,L_1,T), Holds\text{-}at(cond_n,T)$

مجموعه گزاره‌های شماره ۴ برای ساختار شکل شماره ۴

ساختار ۵: بازنشانی ساعت. در برخی حالات سیستم ، باید ساعت‌های محلی و سراسری سیستم بازنشانی شوند تا برخی از حالات سیستم براساس محدودیت‌های ساعت کنترل شود (شکل شماره ۵).

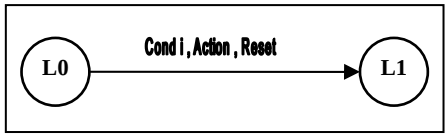
تعریف بصری: در آتاماتای زماندار می توانیم یک تغییر حالت را با درست بودن یک شرط و بعد از انجام یک عملیات داشته باشیم که این تغییر حالت را با قوانین زیر انجام می‌دهیم (با توجه به اینکه هر عمل بازنشانی را می توانیم معادل با یک رخداد در نظر بگیریم).

تولید گزاره برای ساختار ۵: درصورتی، در زمان Ti در حالت L1 قرار داریم که در زمان Ti-1 رخدادهای Action و Start(reset) اتفاق بیافتند (که رخداد Start(reset) نشان دهنده شروع رخداد برای بازنشانی ساعت می‌باشد) و حالت L0 درصورتیکه شرط Condi درهمان زمان برقرار باشد غیر فعال می‌شود و در زمان Ti رخداد End(reset) ساعت را برابر صفر قرار می‌دهد.

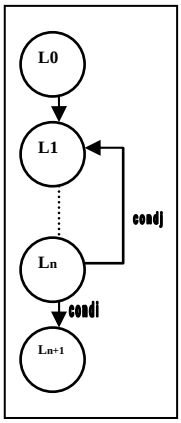
$HoldsAt(L_1,Ti) \leftarrow Happens(Action,T_{i-1}), Happens(Start(reset),T_{i-1}), Terminates(L_0,Action,T_{i-1}), Holds\text{-}at(Condi,T_{i-1}), Happens(End(reset),Ti)$

گزاره شماره ۵ برای ساختار شکل شماره ۵

ساختار ۶: تکرار تناوبی حالات سیستم. در پایش رفتار سیستم بعضی از حالات بصورت دوره‌ای اجرا می‌شوند که در شکل شماره۶ نشان داده شده است.



شکل شماره ۵- بازنشانی ساعت در آتاماتای زماندار با رخداد Reset



شکل شماره ۶- تکرار تناوبی حالات سیستم در آتاماتای‌زماندار

تعریف بصری: در آتاماتای‌زماندار ممکن است گروهی از حالت‌ها و فعالیت‌ها را برای چندین بار اجرا کنیم. بعبارت دیگر در مسیر حرکت حلقه‌ای وجود داشته باشد که حالت‌های شرکت کننده در این حلقه چندین بار اجرا می‌شوند (شکل شماره ۶).

تولیدگزاره برای ساختار ۶: فرض کنید در زمان T1 در حالت L1 قرار داشته باشیم، آنگاه: (۱) اگر در زمان T (T<T1) رخدادی رخ می‌دهد که باعث غیرفعال شدن حالت L0 شود پس در زمان 'T+T در حالت L1 قرار خواهیم گرفت یا (۲) اگر در زمان T رخدادی رخ داده باشد که باعث غیر فعال شدن حالت Ln شده باشد و شرط Condj برقرار باشد، آنگاه در زمان 'T+T در حالت L1 قرار خواهیم گرفت (که در واقع نشان دهنده حلقه بوجود آمده می‌باشد) در غیر اینصورت در صورت برقراری شرط Condi در زمان 'T+T در حالت Ln+1 قرار خواهیم گرفت.

$HoldsAt(L_1,T_1) \leftarrow Happens(e,T), Terminates(e,L_0,T), AntiTrajectory(L_0,T,L_1,T') \quad (T_1>=T+T')$
$HoldsAt(L_1,T_1) \leftarrow Happens(e,T), Terminates(e,L_n,T), AntiTrajectory(L_n,T,L_1,T'), Holds\text{-}at(Cond_j,T)$
$HoldsAt(L_{n+1},T_1) \leftarrow Happens(e,T), Terminates(e,L_n,T), AntiTrajectory(L_n,T,L_{n+1},T'), Holda\text{-}at(Cond_i,T)$

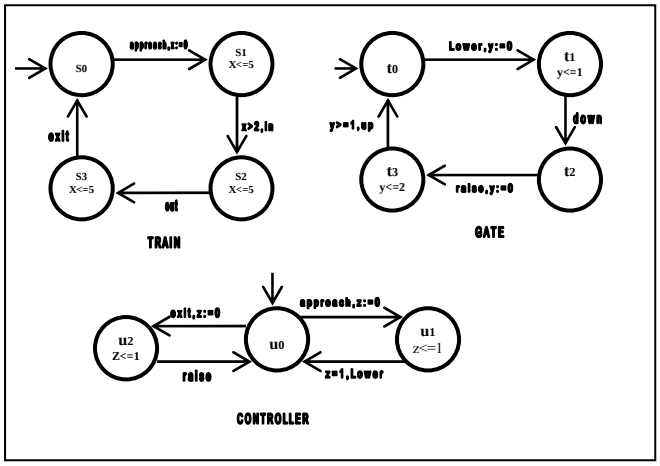
گزاره شماره ۶ برای ساختار شکل شماره ۶

۴-۲- تعیین گراف دسترسی و تولید قوانین راستی‌آزمایی

ما در این بخش با استفاده از گزاره‌هایی که در بخش ۴-۱ تولید کردیم و با استفاده از ساخت گراف دسترسی که در این بخش از روی آتاماتای زماندار می‌سازیم، به تولید قوانین راستی‌آزمایی می‌پردازیم. ما این مراحل را با طرح یک مسئله کنترل تقاطع راه‌آهن و جاده نمایش می‌دهیم. در کنترل کننده خودکار تقاطع راه‌آهن و جاده، مانع(گیت) باعث بسته و باز شدن تقاطع می‌شود. این سیستم از سه مولفه تشکیل می شود: قطار، کنترل‌کننده و مانع که در شکل شماره ۷ نشان داده شده است. رابطه قطار با کنترل کننده بوسیله دو رخداد approach و exit

برقرار می‌شود. رخدادهای in وout نشان دهنده رخداد ورود و خروج قطار از تقاطع راه آهن می‌باشد. لازم است قطار سیگنال approach را در کمتر از دو دقیقه، قبل از آنکه وارد تقاطع شود ارسال کند و این نیاز بوسیله شرط guard ، X>2 به همراه رخداد in نشان داده شده است. بعلاوه می‌دانیم که بیشترین تاخیر بین سیگنال‌های approach وexit پنج دقیقه است و این نیازمند آن است که بوسیله invariant ،X<=5 در حالت‌های S1،S2وS3 نشان داده شود.گیت در حالت t0 باز ودر حالت t2 بسته است؛گیت بوسیله سیگنال‌های Lower وraise با کنترل‌کننده ارتباط دارد. رخدادهای up وdown باز بودن و بسته بودن گیت را مشخص می‌کنند؛ گیت به سیگنال lower با بسته شدن در یک دقیقه و به سیگنال raise با باز شدن در یک تا دو دقیقه جواب می‌دهد. ساعت y جهت نمایش محدودیت‌ها(قیود) استفاده می‌شود.حالت بیکار کنترل‌کننده حالت u0 می‌باشد؛ هر موقع که کنترل‌کننده، سیگنال approach را از قطار دریافت می‌کند آن بوسیله ارسال سیگنال lower به گیت واکنش نشان می‌دهد و هر موقع کنترل کننده سیگنال exit را از قطار دریافت می‌کند بوسیله ارسال سیگنال raise به گیت، واکنش نشان می‌دهد. زمان پاسخ کنترل‌کننده به سیگنال approach یک دقیقه و به سیگنال exit بیش از یک دقیقه می‌باشد؛این محدودیت‌ها بوسیله ساعت z نشان داده شده است.

حالت امن برای سیستم آن است که وقتی قطار داخل تقاطع است گیت باید پایین باشد یعنی در هر حالت دسترس پذیر،اگر قطار در حالت s2 قرار دارد گیت باید در حالت t2 باشد. برای مثال یک لبه از حالت اولیه (s0,t0,u0) به (s1,t0,u1) و از (s1,t0,u1) به (s2,t0,u1) وجود دارد که مطابق با سناریو، رخداد approach فورا بوسیله رخداد in دنبال می‌شود. اگر کمی در مورد اطلاعات زمانی فکر کنیم متوجه می شویم که رخداد approach نمی‌تواند فورا بوسیله رخداد in دنبال شود.



شکل شماره ۷-توصیف بصری رفتار سیستم کنترل تقاطع جاده و ریل آهن با آتاماتای‌زماندار[۳]

حال براساس قوانین مطرح شده در بخش ۴-۱ مدل بالا را با حساب-رخداد شبیه سازی می‌کنیم:

InitiallyP(s_0,t_0,u_0)

توصیف شماره ۱

شرح توصیف شماره۱: در زمان صفر Train وGate وController به ترتیب در حالت s0 وt0 وu0 قرار دارند.

HoldsAt(s_1,T_1) ⟵ Happens(signal(approach),T_0) ,
Happens(Start(reset(x)),T_0) , AntiTrajectory(s_0,T_0,s_1,T'_0) ,
Happens(End(reset(x)),T_1) (T_1=T_0+T'_0)

توصیف شماره ۲

شرح توصیف شماره۲: [**استفاده از گزاره‌های ۳ و ۵]:** قطار در صورتی در زمان T1 درحالتs1 قرار می‌گیرد که در زمان T0 سیگنالی را دریافت کرده باشد (Happens(signal(approach),T0)) و رخداد reset کردن ساعت x هم در همین زمان شروع شده باشد (Happens(Start(reset(x)),T0)) آنگاه در T0+T'0 واحد زمانی بعد در حالت s1 قرار می‌گیرید (AntiTrajectory(s0,T0,s1,T'0)) وسپس زمانیکه سیستم در حال وارد شدن به حالت s1 است مقدار ساعت x با رخداد Happens(End(reset(x)),T1) صفر می‌شود.

HoldsAt(s_1,T_2)⟵ HoldsAt(s_1,T_1) ,
Holds-at (cond (x<=5),T_2)

توصیف شماره ۳

شرح (توصیف شماره۳: **[استفاده از گزاره۲]:** قطار تا زمانیکه شرط x<=5 ،در حالت s1 برقرار باشد می‌تواند در این حالت باقی بماند.(در واقع دستورات بالا نشان دهنده آنست که اگر قطار در زمان T1در حالت s1 قرار داشته باشد در صورتی در زمان T2 در همان حالت باقی می-ماند که شرط x<=5 برقرار باشد.)

HoldsAt(u_1,T_1) ⟵ Happens(signal(approach),T_0)
,Happens(Start(reset(z)),T_0) , AntiTrajectory(u_0,T_0,u_1,T'_0)
, Happens(End(reset(z)),T_1) (T_1=T_0+T'_0)

توصیف شماره ۴

شرح توصیف شماره۴: [**استفاده از گزاره‌های۳ و ۵]:**کنترل‌کننده در صورتی در زمان T1 در حالت u1 قرار می‌گیرد که رخداد دریافت سیگنال که مبنی بر تغییر حالت قطار می‌باشد را در زمان T0 دریافت کند و رخداد reset شدن ساعت نیز در همین زمان شروع شود آنگاه در T0+T'0 واحد زمانی بعد در حالت u1 قرار می‌گیرد و در حالی که در زمان T1 وارد حالت u1 می‌شود رخداد Happens(End(reset(z)),T1) باعث صفر شدن ساعت z می‌شود.

$HoldsAt(u_1,T_1) \longleftarrow HoldsAt(u_1,T_0)$,
$Holds\text{-}at\ (cond\ (z<=1),T_1)$

توصیف شماره ۵

شرح توصیف شماره۵: [**استفاده از گزاره ۲] :**کنترل کننده تا زمانیکه شرط z<=1 ، در حالت u1 برقرار باشد می‌تواند در این حالت باقی بماند.(در واقع دستورات بالا نشان دهنده آنست که اگر کنترل کننده در زمان T0 در حالت u1 قرار داشته باشد در صورتی، در زمان T1 در همان حالت باقی می‌ماند که شرط z<=1 برقرار باشد.)

$HoldsAt(u_0,t_1,T_3) \longleftarrow Happense(signal(lower),T_2)$,
$Happens(Start(reset(y)),T_2)$, $AntiTrajectory(u_1,T_2,u_0,T'_2)$,
$Holds\text{-}at(cond(z=1),T_2)$, $AntiTrajectory(t_0,T_2,t_1,T'_2)$,
$Happens(End(reset(y)),T_3)$. $(T_2+T'_2=T_3)$

توصیف شماره ۶

شرح توصیف شماره ۶: [**استفاده از گزاره‌های ۳ و ۴ و ۵] :**در زمان T3 کنترل‌کننده در حالت u0 و گیت در حالت t1 قرار خواهد گرفت در صورتیکه سیگنال lower در زمان T2 اتفاق بیافتد و رخداد reset کردن ساعت y هم در همین زمان شروع شود آنگاه در T2+T'2 واحد زمانی بعد گیت وارد حالت t1 و در صورتیکه شرط z=1 برقرار باشد کنترل‌کننده هم وارد حالت u0 می‌شود و در حالی که در زمان T3 گیت وارد حالت t1 می‌شود رخداد Happens(End(reset(y)),T3) باعث صفر شدن ساعت y می‌شود.

$HoldsAt(t_1,T_4) \longleftarrow HoldsAt(t_1,T_3)$,
$holds\text{-}at\ (cond\ (y<=1),T_4)$

توصیف شماره ۷

شرح توصیف شماره ۷: [**استفاده از گزاره‌های ۲ و ۴] :**گیت تا زمانیکه شرط y<=1 ،در حالت t1 برقرار باشد می‌تواند در این حالت باقی بماند.(در واقع دستور بالا نشان دهنده آنست که اگر گیت در زمان T3در حالت t1 قرار داشته باشد در صورتی، در زمان T4 در همان حالت باقی می‌ماند که شرط y<=1 برقرار باشد.)

$HoldsAt(t_2,T_6) \longleftarrow Happens(down,T_5)$,
$AntiTrajectory(t_1,T_5,t_2,T'_5)$ $(T_5+T'_5=T_6)$

توصیف شماره ۸

شرح توصیف شماره ۸: [**استفاده از گزاره ۳] :**گیت در زمان T6 در حالت t2 قرار خواهد گرفت در صورتیکه رخداد down در زمان T5 اتفاق بیافتد و در اثر این رخداد گیت در T5+T'5 واحد زمانی بعد وارد حالت t2 می‌شود.

$HoldsAt(s_2,T_8) \longleftarrow Happens(in,T_7)$,
$Holds\text{-}at(cond(x>2),T_7)$, $AntiTrajectory(s_1,T_7,s_2,T'_7)$,
$Holds\text{-}at(cond(x<=5),T_7+T'_7)$ $(T_8=T_7+T'_7)$

توصیف شماره ۹

شرح توصیف شماره ۹: [**استفاده از گزاره‌های ۳و۴] :**قطار در زمان T8در حالت s2 قرار خواهد گرفت در صورتیکه رخداد in در زمان T7 اتفاق بیافتد و شرط ساعت x>2 هم در زمان T7 برقرار باشد آنگاه قطار در T7+T'7 واحد زمانی بعد در حالت s2 قرار خواهد گرفت در صورتیکه در همین زمان شرط x<=5 هم برقرار باشد.

$HoldsAt(s_2,T_9) \longleftarrow HoldsAt(s_2,T_8)$,
$Holds\text{-}at\ (cond\ (x<=5),T_9)$

توصیف شماره ۱۰

شرح توصیف شماره ۱۰: [**استفاده از گزاره ۲] :**قطار تا زمانیکه شرط x<=5 ، در حالت s2 برقرار باشد می‌تواند در این حالت باقی بماند.(در واقع دستور بالا نشان دهنده آنست که اگر قطار در زمان T8در حالت s2 قرار داشته باشد در صورتی، در زمان T9 در همان حالت باقی می‌ماند که شرط x<=5 برقرار باشد.)

$HoldsAt(s_3,T_{11}) \longleftarrow Happens(out,T_{10})$,
$AntiTrajectory(s_2,T_{10},s_3,T'_{10})$, $Holds\text{-}at\ (cond\ (x<=5),$
$T_{10}+T'_{10})$ $(T_{11}=T_{10}+T'_{10})$

توصیف شماره ۱۱

شرح توصیف شماره ۱۱: [**استفاده از گزاره‌های ۳و۴] :**قطار در زمان T11 در حالت s3 قرار خواهد گرفت در صورتیکه رخداد out در زمان T10 اتفاق افتاده باشد آنگاه در T10+T'10 واحد زمانی بعد در صورتیکه شرط x<=5 برقرار باشد وارد حالت s3 می‌شود.

$HoldsAt(s_0,T_{13}) \longleftarrow Happens(signal(exit),T_{12})$,
$AntiTrajectory(s_3,T_{12},s_0,T'_{12})$ $(T_{13}=T_{12}+T'_{12})$

توصیف شماره ۱۲

شرح توصیف شماره ۱۲:[**استفاده از گزاره ۳] :**قطار در زمان T13 در حالت s0 قرار خواهد گرفت در صورتیکه در زمان T12 سیگنال exit که مبنی بر خروج قطار از تقاطع می‌باشد را دریافت کرده آنگاه در T12+T'12 واحد زمانی بعد در حالت s0 قرار می‌گیرد.

$HoldsAt(u_2,T_{13}) \longleftarrow Happens(signal(exit),T_{12})$,
$Happens(Start(reset(z)),T_{12})$,
$AntiTrajectory(u_0,T_{12},u_2,T'_{12})$,
$Happens(End(reset(z)),T_{13})$ $(T_{13}=T_{12}+T'_{12})$

توصیف شماره ۱۳

شرح توصیف شماره ۱۳: [**استفاده از گزاره‌های ۳و۵] :**کنترل‌کننده در زمان T13در حالت u2 قرار خواهد گرفت در صورتیکه سیگنال exit که مبنی بر خارج شدن قطار از تقاطع است را در زمان T12 دریافت کند و در همین زمان رخداد reset شدن ساعت z هم شروع شود آنگاه در T12+T'12 واحد زمانی بعد وارد حالت u2 می‌شود ودر همین زمان مقدار ساعت z صفر می‌شود.

$HoldsAt(u_2,T_{15}) \longleftarrow HoldsAt(u_2,T_{14})$,
$Holds\text{-}at\ (cond\ (z<=1),T_{15})$

توصیف شماره ۱۴

شرح توصیف شماره ۱۴: **استفاده از گزاره‌های ۲و۴]** :کنترل‌کننده تا زمانیکه شرط z<=1 ، در حالت u2 برقرار باشد می‌تواند در این حالت باقی بماند.(در واقع دستور بالا نشان دهنده آنست که اگر کنترل‌کننده در زمان T14در حالت u2 قرار داشته باشد در صورتی، در زمان T15 در همان حالت باقی می‌ماند که شرط z<=1 برقرار باشد.)

$$HoldsAt(u_0,T_{17}) \longleftarrow Happens(signal(raise),T_{16}),\ AntiTrajectory(u_2,T_{16},u_0,T'_{16}) \quad (T_{17}=T_{16}+T'_{16})$$

توصیف شماره ۱۵

شرح توصیف شماره ۱۵: **استفاده از گزاره ۳]** :کنترل‌کننده در زمان T17 در حالت u0 قرار خواهد گرفت در صورتیکه سیگنال raise را در زمان T16 دریافت کرده آنگاه در T16+T'16 واحد زمانی بعد در حالت u0 قرار می‌گیرد.

$$HoldsAt(t_3,T_{17}) \longleftarrow Happens(signal(raise),T_{16}),\ Happens(Start(reset(y)),T_{16}),\ AntiTrajectory(t_2,T_{16},t_3,T'_{16}),\ Happens(End(reset(y)),T_{16}+T'_{16}),\ Holds\text{-}at(Cond(y<=2),T_{16}+T'_{16}) \quad (T_{17}=T_{16}+T'_{16})$$

توصیف شماره ۱۶

شرح توصیف شماره ۱۶: [**استفاده از گزاره‌های ۳و۴و۵**] :گیت در زمان T17 در حالت t3 قرار خواهد گرفت در صورتیکه سیگنال raise را در زمان T16 دریافت کند و در همین زمان رخداد reset شدن ساعت y شروع شود آنگاه در T16+T'16 واحد زمانی بعد در صورتیکه شرط y<=2 برقرار باشد وارد حالت t3 می‌شود و در همین زمان رخداد Happens(End(reset(y)),T16+T'16) باعث صفر شدن ساعت y می‌شود.

$$HoldsAt(t_3,T_{18}) \longleftarrow HoldsAt(t_3,T_{17}),\ Holds\text{-}at\ (cond\ (y<=2),T_{18})$$

توصیف شماره ۱۷

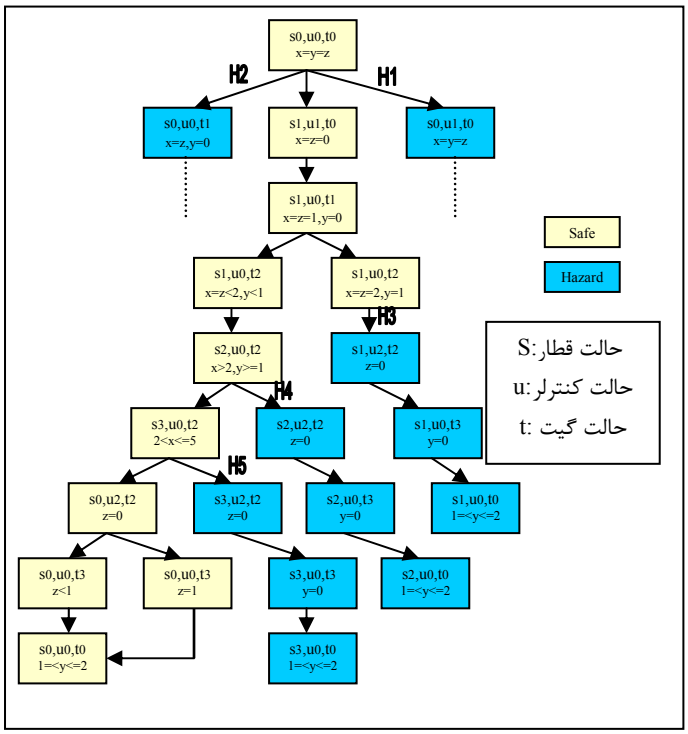
شرح توصیف شماره ۱۷:[**استفاده از گزاره‌های ۲و۴**] :گیت تا زمانیکه شرط y<=2 ، در حالت t3 برقرار باشد می‌تواند در این حالت باقی بماند.(در واقع دستور بالا نشان دهنده آنست که اگر گیت در زمان T17 در حالت t3 قرار داشته باشد در صورتی، در زمان T18 در همان حالت باقی می‌ماند که شرط y<=2 برقرار باشد.)

$$HoldsAt(t_0,T_{20}) \longleftarrow Happens(up,T_{19}),\ Holds\text{-}at\ (cond\ (y<=2),T_{19}),Holds\text{-}at(cond(y>=1),T_{19}),\ AntiTrajectory(t_3,T_{19},t_0,T'_{19}) \quad (T_{20}=T_{19}+T'_{19})$$

توصیف شماره ۱۸

شرح توصیف شماره ۱۸[**استفاده از گزاره‌های ۳و۴]** :گیت در زمان T20 در حالت اولیه خود t0 قرار خواهد گرفت در صورتیکه در زمان T19 رخداد up اتفاق بیافتد و در همین زمان شرط‌های y<=2 و y>=1 بر قرار باشند آنگاه در T19+T'19 واحد زمانی بعد وارد حالت t0 می‌شویم.

۴-۳- تعیین گراف دسترسی: همان طور که در بالا مشاهده کردید ما مسئله تقاطع راه آهن و جاده را که با مدل آتاماتای‌زماندار پیاده سازی شده بود را براساس قوانین مطرح شده در قسمت رویکرد پیشنهادی، با حساب‌رخداد مدل سازی کردیم و از آنجایی که مدل‌های گرافیکی به خوبی نمی‌توانند قیود موجود در سیستم را پیاده سازی کنند به همین خاطر به یک زبان مدل‌سازی متنی جهت بیان قیود نیاز دارند، ما می‌توانیم رفتار کلی سیستم را بوسیله آتاماتای‌زماندار نشان دهیم و سپس همه ترکیب‌های حالات سیستم و محیط را بوسیله گراف reachability نشان دهیم و برای آن ترکیب‌هایی را که باعث می‌شود سیستم وارد حالت‌های ناامن شود، بوسیله حساب قانون تولید کنیم. در شکل شماره۸، گراف reachability مسئله تقاطع راه‌آهن و جاده را نشان می‌دهیم و سپس قوانین راستی‌آزمایی را با حساب‌رخداد تولید می‌کنیم.



شکل شماره ۸-گراف دسترسی مثال تقاطع راه‌آهن و جاده [شکل۷]

قانون راستی‌آزمایی ۱

$$HoldsAt\ (s_1,T) \longleftarrow HoldsAt\ (u_0,T)\ ,HoldsAt\ (t_1,T)$$

شرح: زمانیکه قطار به تقاطع نزدیک می‌شود(approach) گیت باید در حال پایین آمدن(t1) باشد.

قانون راستی‌آزمایی ۲

$$HoldsAt(s_1,T) \longleftarrow HoldsAt(u_0,T)\ ,\ (\neg HoldsAt(t_0,T)\ OR\ \neg HoldsAt(t3,T))$$

شرح: زمانیکه قطار به تقاطع نزدیک می‌شود (approach) گیت نباید بالا یا در حال بالا رفتن(t0,t3) باشد(کنترل‌کننده حالت ناامن H3 می‌باشد).

قانون راستی‌آزمایی ۳

HoldsAt(s2,T)⟵ HoldsAt(u0,T), HoldsAt(t2,T)

شرح: زمانیکه قطار در تقاطع قرار دارد(s2) گیت باید پایین باشد(t2) (کنترل‌کننده حالت ناامن H4 می‌باشد).

قانون راستی‌آزمایی ۴

HoldsAt(s2,T)⟵ HoldsAt(u0.T), (¬ HoldsAt (t1,T) OR ¬HoldsAt(t0,T))

شرح: زمانیکه قطار در تقاطع قرار دارد(s2) گیت نباید بالا یا در حال پایین آمدن باشد(کنترل‌کننده حالت ناامن H4 و H5 می‌باشد).

قانون راستی‌آزمایی ۵

HoldsAt(s3,T)⟵ HoldsAt(u0,T), HoldsAt(t2,T)

شرح: زمانیکه قطار تقاطع را بطور کامل ترک نکرده (s3) گیت باید در حالت پایین باشد. (کنترل‌کننده حالت ناامن H5 می‌باشد).

قانون راستی‌آزمایی۶

HoldsAt(s3,T)⟵ HoldsAt(u0,T), (¬HoldsAt(t1,T) OR ¬HoldsAt(t0,T))

شرح: زمانیکه قطار تقاطع را بطور کامل ترک نکرده (s3) گیت نباید بالا یا در حال پایین آمدن باشد(کنترل‌کننده حالت ناامن H5 می‌باشد).

قانون راستی‌آزمایی ۷

HoldsAt(s0,T)⟵ HoldsAt(u0,T),HoldsAt(t3,T)

شرح: زمانیکه قطار تقاطع را بطور کامل ترک می‌کند گیت باید در حال بالا آمدن باشد. .(کنترل‌کننده حالت ناامن H2 می‌باشد).

قانون راستی‌آزمایی ۸

HoldsAt(s0,T) ⟵HoldsAt(u0,T), (¬HoldsAt(t1,T) OR ¬HoldsAt(t2,T))

شرح: زمانیکه قطار در تقاطع وجود ندارد گیت نباید پایین یا در حال پایین آمدن باشد(کنترل‌کننده حالت ناامن H1 و H2 می‌باشد).

۵- نتیجه‌گیری

ما در این مقاله روشی را ارائه دادیم که به وسیله آن می‌توان از توصیف یک مسئله با آتاماتای زمانی، قوانینی را تولید کرد که حالت‌های مخاطره‌آمیز ممکن را راستی‌آزمایی کند. این قوانین هم در راستی-آزمایی توصیف مسئله و هم در راستی‌آزمایی پیاده‌سازی آن به ما کمک می‌کند. آنچه که تولید قوانین راستی‌آزمایی را از توصیف‌های آتاماتای زمانی میسر ساخت، منطق زمانی حساب رخداد بود. این حساب از یک طرف به دلیل این که دارای روش‌های استنتاجی است به ما کمک می-کند تا قوانین راستی‌آزمایی حاصل را برای بررسی درستی توصیف مسئله به کاربریم و از طرف دیگر می‌توان از آن برای راستی‌آزمایی پیاده‌سازی توصیف استفاده نمود. استفاده از این حساب که هم در توصیف آتاماتا و هم در گراف حالت‌های دسترسی آن استفاده شد، کاستی روش‌های توصیف بصری را در راستی‌آزمایی توصیف جبران کرد. استخراج گزاره‌های حساب رخداد از توصیف‌های بصری شبکه‌های پتری [۸]، نمونه دیگری از تولید قوانین راستی‌آزمایی با استفاده از حساب رخداد است که برای جبران کاستی روش‌های توصیف بصری در راستی-آزمایی به کار گرفته شده است. حساب رخداد که در این مقاله برای سیستم‌های زمان واقعی به کار گرفته شد در [۹] برای تولید قوانین راستی‌آزمایی در سیستم‌های توزیع‌شده نیز به کار گرفته شده است.

مراجع:

[1] Alur, R., Dill, D.L., "Automata-theoretic verification of real-time systems", Formal Methods for Real-Time Computing, Trends in Software Series, John Wiley & Sons Publishers, pp. 55-82, 1996.

[2] Kawalski, R.A., Sergot, M.J., "A Logic Based Calculus of Events", New Generation Computing, Vol. 4, 67-95, 1986.

[3] Krákora, J., Hanzálek, Z., "Timed Automata Approach for CAN Verification", 11th IFAC Symposium on Information Control Problems in Manufacturing, INCOM, Salvador, ElsevierApril 2004.

[4] Waszniowski, L. , Hanzalek, Z., "Over-approximate Model of Multitasking Application Based on Timed Automata Using Only One Clock", 19th IEEE International Parallel and Distributed Processing Symposium IPDPS 2005, WPDRTS, Denver, USA, April, 2005.

[5] Krákora, J. , Hanzálek, Z., "Optimisation of Applications for FPGAs with PowerPC Processor Using Priced Timed Automata", In IEEE International Symposium on Industrial Electronics, ISIE 2008, Cambridge: Anglia Ruskin University, 2008.

[6] Krákora, J., Waszniowski, L., Píša P., Hanzálek, Z., "Timed Automata Approach to Real Time Distributed System Verification", 5th IEEE International Workshop on Factory Communication Systems, WFCS, Vienna, September 22-24, 2004.

[7] Sadri, F. and Kowalski, R., Variants of the Event Calculus, In Proceedings of the 12th International Conference on Logic Programming (ICLP), MIT Press, pp. 67-82, 1995.

[8]S.M. Babamir and F.S. Babamir, Behavioral Specification of Real-time Requirements, 15th Asia-Pacific Software Engineering Conference, 2008, China.

[۹] سید مرتضی بابامیر و سعید جلیلی. رویکردی برای وارسی پویا و مبتنی بر منطق سیستم‌های توزیع‌شده، یازدهمین کنفرانس بین المللی انجمن کامپیوتر ایران، پژوهشگاه دانشهای بنیادی، ۱۳۸۴.