# چهارچوبی برای تشخیص خاتمه پذیری قوانین فعال با استفاده از وارسی مدل مبتنی بر BDD و SAT

**نعمت‌اله کمال فر**، دانشگاه آزاد اسلامی مشهد - گروه مهندسی کامپیوتر – n.kamalfar@mshdiau.ac.ir
**مسعود نیازی ترشیز**، دانشگاه New South Wales استرالیا-گروه علوم کامپیوتر– masoodt@cse.unsw.edu.au
**مجید وفایی جهان**، دانشگاه آزاد اسلامی مشهد - گروه مهندسی کامپیوتر – vafaeijahan@mshdiau.ac.ir

**چکیده:** یکی از مهم‌ترین مشکلات مربوط به قوانین فعال، مسئله خاتمه ناپذیری آن‌هاست. تاکنون روش‌های متعددی برای بررسی خاتمه پذیری قوانین فعال پیشنهاد شده‌اند. بسیاری از روش‌های پیشنهادی، محافظه‌کارانه عمل نموده و قادر به اظهار نظر قطعی در مورد خاتمه ناپذیری قوانین نمی‌باشند. در این میان چند روش نیز با بهره‌گیری از وارسی مدل، موفق به آنالیز خودکار و غیر محافظه‌کارانهٔ خصوصیت خاتمه‌پذیری قوانین فعال گشته‌اند. اما این روش‌ها نیز از مشکل اصلی اکثر روش‌های مبتنی بر وارسی مدل، یعنی انفجار فضای حالت رنج می‌برند. همچنین اکثر روش‌های پیشنهادی در گذشته، از استراتژی‌های مختلف پردازش قوانین پشتیبانی نمی‌نمایند.
ما در روش پیشنهادی خود به منظور اجتناب از مشکل انفجار فضای حالت، از وارسی مدل مبتنی بر BDD و SAT بهره گرفته‌ایم. بر اساس چهارچوب پیشنهادی ما، خصوصیت خاتمه پذیری قوانین فعال تحت استراتژی‌های مختلف پردازش قوانین و با استفاده از وارس مدل قدرتمندی به نام NuSMV۲ قابل بررسی می‌باشد. بعلاوه، چهارچوب پیشنهادی به دلیل عدم وابستگی ساختاری به معماری پایگاه داده فعال، به راحتی برای تمامی سیستم‌های مبتنی بر قوانین فعال قابل تطبیق و استفاده است.

**کلمات کلیدی:** قوانین فعال، پایگاه داده فعال، خاتمه پذیری، وارسی مدل

## ۱- مقدمه

قوانین فعال که به قوانین رویداد-شرط-عمل[۱] یا قوانین ECA نیز مشهورند، امکان واکنش شرطی نسبت به رویدادهای محیطی را برای سیستم‌های تحریک شونده با رویداد فراهم می‌آورند. این قوانین می‌توانند نسبت به هر ترکیبی از رخدادها، در هر زمان و با هر ترتیب، واکنش نشان دهند. کارایی و توانایی‌های قوانین فعال موجب جلب توجه روز افزون محققین در شاخه های مختلف سیستم‌های پایگاه داده، سیستم‌های مبتنی بر دانش، و دیگر سیستم‌های تحریک شونده با رویداد، به سوی این قوانین شده است. از جمله این زمینه های متنوع کاربرد قوانین فعال می‌توان به پایگاه داده حسگری[۱] ،وب معنایی[۲] ،XML [۳]، سیستم‌های انبارسازی داده‌ها[۴] و سیستم‌های تجاری انطباق پذیر [۵] اشاره نمود. یکی از معمول‌ترین و مهم‌ترین زمینه های کاربرد قوانین فعال، سیستم‌های پایگاه داده فعال[۲] می‌باشد [۶]. سیستم‌های پایگاه داده می‌توانند به کمک قوانین فعال نسبت به رویدادهای خارجی، داخلی، و حتی ترکیبی از آن‌ها واکنش نشان دهند. درحالی‌که پایگاه داده های سنتی تنها قادر به واکنش نسبت به رویدادهای خارجی بودند.

استفاده از قوانین در چنین سیستم‌هایی نیازمند آنالیز کامل و دقیق رفتار قوانین می‌باشد. اما تعامل و رفتار واکنشی موجود بین قوانین، آنالیز رفتاری مجموعه ای از قوانین فعال را به کاری دشوار و پیچیده بدل می‌کند [۷]. مهم‌ترین و معمول‌ترین مشکل رفتاری قوانین فعال، مسئله «خاتمه ناپذیری» آن‌ها می‌باشد. اجرای هر قانون ممکن است موجب تحریک قانونی دیگر شده و زنجیره ای نامتناهی از تحریک قوانین را به دنبال داشته باشد. در چنین شرایطی فرآیند پردازش قوانین پایان نخواهد یافت و مجموعه قوانین مربوطه خاتمه ناپذیر خوانده می‌شوند.

در ادامه و در بخش۲ مروری خواهیم داشت بر کارهایی که پیش از این در زمینه تحلیل خاتمه پذیری قوانین فعال صورت گرفته‌اند. سپس به تشریح چهارچوب پیشنهادی خود پرداخته، خواهیم گفت که سیستم نمونه چگونه مدل سازی شده و وارسی مدل بر روی آن انجام می‌شود. در نهایت نیز نتایج بدست آمده را ارائه نموده و با روش پیشین مقایسه خواهیم نمود.

---

[۱] Event-condition-action
[۲] Active database

## ۲- کارهای مرتبط

### ۲-۱- روش‌های ایستا

اکثر تحقیقات پیشین صورت گرفته در زمینه آنالیز رفتاری قوانین فعال، از روش‌های ایستا بهره گرفته‌اند [۸] [۷] [۹] [۱۰] . یک روش ایستا سعی دارد تا با تحلیل مجموعه‌ی قوانین پیش از اجرای آن‌ها، خاتمه پذیر بودنشان را تایید و اثبات نماید. تحلیل ایستا تاکنون با استفاده از روش‌های مختلفی صورت گرفته است. استفاده از یک زبان توصیف قوانین همچون Starburst [۸]، استفاده از جبر رابطه ای [۱۰]، و استفاده از شبکه های پتری[۳] [۹]، از جمله این روش‌ها می‌باشند. مشکل اصلی روش‌های ایستا، محافظه کارانه بودن آن‌هاست. این روش‌ها قادر به اظهار نظر قطعی در زمینه خاتمه ناپذیری قوانین نبوده، نیازمند دخالت طراح سیستم برای آزمایش دستی قوانین مربوطه و یا استفاده از یک روش مکمل می‌باشند.

### ۲-۲- روش‌های مبتنی بر وارسی مدل[۴]

وارسی مدل از نظر مفهومی چنین است که تمامی حالات ممکن از مدل سیستم تولید شده و بررسی می‌شود که آیا در هر حالت مشخصهٔ مورد نظر برقرار می‌باشد یا خیر [۱۱] [۱۲]. استفاده از وارسی مدل از یک سو امکان بررسی صحت رفتاری قوانین در فاز طراحی و توسعه سیستم را فراهم آورده، از سوی دیگر مشکلات مربوط به روش‌های ایستا را به دنبال نخواهد داشت.

اولین بار در سال ۱۹۹۸ طی تحقیقی در زمینه ارائه یک روش خودکار برای تحلیل و درستی یابی سیستم‌های مدیریت پایگاه داده فعال، از وارسی مدل استفاده شد [۱۳]. چندی بعد در سال ۲۰۰۱، روشی مبتنی بر وارسی مدل نمادین برای تشخیص خاتمه پذیری قوانین پایگاه داده فعال ارائه شد [۱۴]. این روش همان‌گونه که خود نویسنده می‌گوید، روشی تکمیلی برای استفاده در کنار روش‌های ایستای محافظه کارانه است. از جمله مشکلات این روش می‌توان به عدم توانایی در شبیه سازی صحیح رفتار سیستم و عدم مدل سازی تراکنش‌ها اشاره نمود. لازم به ذکر است که روش‌های پیشنهادی در [۱۳]و [۱۴] از استراتژی‌های مختلف پردازش قوانین پشتیبانی نمی‌نماید.

در سال ۲۰۰۶، سه محقق ژاپنی اقدام به تکمیل و بهبود چهارچوب ارائه شده در [۱۳] نمودند [۱۵]. چهارچوب ارائه شده توسط آن‌ها، همچون روش قبلی در محیط Spin [۱۶] پیاده سازی شده و مبتنی بر وارسی مدل حالت-صریح می‌باشد. با وجود اینکه مدل پیشنهادی آن‌ها از بسیاری جهات کاملاً مطابق مدل قبلی می‌باشد، به شکلی ساخت یافته تر پیاده سازی شده و از مزیت پشتیبانی از استراتژی‌های مختلف پردازش قوانین برخوردار است. چهارچوب پیشنهادی آن‌ها، کامل‌ترین و کارآمد روش مبتنی بر وارسی مدل برای بررسی خاتمه پذیری در سیستم‌های پایگاه داده فعال می‌باشد که پیش از این ارائه شده است. اما این روش از مشکلی شناخته شده بین تکنیک‌های وارسی مدل، یعنی مشکل «انفجار فضای حالت[۵]» رنج می‌برد. انفجار فضای حالت زمانی وقوع می‌یابد که فضای حالت ایجاد شده از روی مدل بیش از حد بزرگ بوده و در حافظه سیستم جای نگیرد. به طور کلی تکنیک وارسی مدل حالت-صریح در مقابله با این مشکل عملکرد مناسبی ندارد.

## ۳- روش پیشنهادی

چنانچه اشاره شد، مشکل اصلی روش‌های مبتنی بر وارسی مدل، مسئله انفجار فضای حالت می‌باشد. این مشکل موجب شده است روشی همچون [۱۵] که بهترین کار پیشین انجام شده بر اساس وارسی مدل می‌باشد، عملاً به روشی ناکارآمد در بررسی صحت رفتاری قوانین سیستم‌های حقیقی بدل شود. هدف ما ارائه روش و چهارچوبی ست که علیرغم بهره گیری از مزایای وارسی مدل، به خوبی امکان اجتناب از مشکل انفجار فضای حالت را فراهم آورده و برای سیستم‌های حقیقی قابل استفاده باشد. بدین منظور بهترین روش موجود

---

[۳] Petri-nets
[۴] Model Checking
[۵] State-space explosion

برای مقابله با مشکل انفجار فضای حالت، یعنی وارسی مدل مبتنی بر BDD و SAT را مورد استفاده قرار داده و از یک وارس مدل[۶] به نام NuSMV۲ [۱۷] بهره گرفته‌ایم. این وارس مدل قدرتمند، نتیجه یک پروژه بزرگ با همکاری دانشگاه های مطرح در زمینه وارسی مدل می‌باشد و با حضور مک میلان[۷]، بنیان گذار وارسی مدل نمادین [۱۸]، توسعه یافته است. ما در انجام آزمایشات بر روی روش خود، از دو حل کننده SAT قدرتمند zChaff [۱۹] و MiniSat [۲۰] که امکان اتصال و استفاده از آن‌ها با NuSMV۲ فراهم می‌باشد استفاده نموده‌ایم.

### ۳-۱- سیستم نمونه

از آنجا که تقریباً تمامی کارهای گذشته بر اساس سیستم پایگاه داده نمونه واحدی بنا نهاده شده‌اند، لازم است که ما نیز از نمونه مشابهی استفاده نماییم تا در نهایت بتوانیم مقایسه ای عادلانه داشته باشیم. بنابراین در مدل سازی سیستم، از مجموعه قوانین مورد استفاده توسط روش‌های پیشنهادی گذشته بهره گرفته‌ایم.

پایگاه داده فعال نمونه شامل دو جدول می‌باشد. جدول Emp که از سه رکورد شناسه کارمند (id)، رتبه(rank) و حقوق (salary) کارمند تشکیل یافته است و جدول Bonus که متشکل از رکوردهای شناسه کارمند (id) و پاداش (bonus) کارمند می‌باشد. عملیات‌های داده ای (پرس‌وجو های) مورد نظر ما بر روی این پایگاه داده عبارتند از UpdateRank و UpdateBonus که به ترتیب رتبه و پاداش کارمند را ۱ و ۱۰ واحد افزایش می‌دهند. این دستورات در قالب تراکنش‌هایی به سیستم پایگاه داده ارسال می‌شوند که دربرگیرنده‌ی تعداد مشخصی عملیات داده ای می‌باشند. دو قانون نمونه برای مجموعه قوانین این سیستم پایگاه داده فعال تعریف می‌کنیم. قانون اول bonus-rank نام دارد و مشخص می‌کند که هرگاه پاداش کارمند ۱۰ واحد افزایش یابد، رتبه او ۱ واحد ارتقا می‌یابد. دومین قانون با نام bonus-rank تعیین می‌کند که هر زمان رتبه کارمند به یک مقدار زوج بروز رسانی شود، ۱۰ واحد پاداش به او تعلق خواهد گرفت. مشخصات کامل این پایگاه داده نمونه به همراه تعاریف قوانین به زبان‌های استاندارد، در [۸] و [۱۵] قابل بررسی می‌باشد.

همچون روش‌های پیشنهادی گذشته [۸][۱۳] [۱۴] ، ما نیز برای سادگی فرض می‌کنیم که رویدادهای وقوع یافته در این سیستم پایگاه داده فقط از نوع دست‌کاری داده‌ها هستند، همچنین تراکنش‌ها و قوانین به صورت ترتیبی اجرا می‌شوند.

### ۳-۲- مدل سازی سیستم

از یک سو، NuSMV و اساساً تمامی وارس مدل‌ها برای انجام وارسی مدل نیاز به یک مدل با تعداد حالات متناهی داشته، و برای اجتناب از انفجار فضای حالت لازم است سعی شود که تعداد حالات مدل کمینه باشد. از سوی دیگر چنانچه سیستم مورد نظر مانند این حالت یک سیستم پایگاه داده باشد، ممکن است تعداد حالاتی نامتناهی داشته باشد. بنابراین مهم‌ترین و دشوارترین کاری که باید انجام شود، کوچک نمودن مدل سیستم است به نحوی که رفتار و خصوصیات سیستم دست‌خوش تغییر نشود. در ادامه خواهیم دید که ما چگونه سعی نموده‌ایم سیستم را به بهینه‌ترین شکل ممکن مدل سازی کنیم.

#### ۳-۲-۱ تعریف متغیرهای حالت[۸]

برای تعریف متغیرهای حالت جداول پایگاه داده، به ازای هر فیلد دو متغیر حالت در نظر می‌گیریم، یکی برای حالت جاری و دیگری برای مقدار قبلی آن. اولین قدم و ساده‌ترین راهکار برای کوچک نمودن مدل، حذف متغیرهایی ست که عملاً بلا استفاده‌اند. متغیرهای id، empid و salary اگر چه در سیستم اصلی مورد نیاز می‌باشند، اما تأثیری بر قوانین مورد بررسی ما ندارند، بنابراین از تعریف آن‌ها خودداری می‌شود. بنابراین تنها لازم است متغیرهای rank و bonus را تعریف نماییم. مقادیر این متغیرها در سیستم نمونه نامحدود در

[۶] Model checker
[۷] McMillan
[۸] State variables

نظر گرفته شده است. ما برای شبیه سازی این رفتار و در عین حال اجتناب از انفجار فضای حالت، دامنه این دو متغیر را محدود نموده و برای تغییر حالتشان از عملگر mod (باقیمانده تقسیم) به منظور ایجاد یک روند چرخشی نا محدود استفاده نموده‌ایم.

```
rank : ۰..۴;        bonus : ۰..۲;
oldrank : ۰..۴;     oldbonus : ۰..۲;
```

سیستم نمونه دارای دو قانون فعال می‌باشد که لازم است به نحوی تحریک شدن آن‌ها را در مدلی که ایجاد می‌کنیم مشخص نماییم. بدین منظور، به ازای هر قانون یک متغیر عددی تعریف می‌کنیم.

```
rank-bonus : ۰..۲;
bonus-rank : ۰..۲;
```

استراتژی پردازش قوانین مشخص می‌کند که سیستم چه موقع بایستی به پردازش تراکنش‌های ورودی مشغول باشد و چه هنگام به پردازش قوانین تحریک شده بپردازد. در ادامه خواهیم دید که این استراتژی‌های پردازش قوانین چگونه در چهارچوب پیشنهادی ما مدل سازی می‌شوند. چهار متغیر input، activeRule، transactionSize و ruleProcessing به صورت زیر تعریف شده‌اند تا زیر ساخت لازم را برای این منظور فراهم آورند.

```
input : {uR, uB, n};
activeRule : {rB, bR, n};
ruleProcessing : boolean;
transactionSize : ۰..۲;
```

متغیر input که از نوع شمارشی تعریف شده است، مشخص می‌کند که کدام تراکنش یا دستور ورودی بایستی مورد پردازش قرار گیرد. متغیر ruleProcessing آغاز و پایان فرآیند پردازش قوانین فعال را مشخص نموده و متغیر activeRule، قانون فعال و تحت پردازش را تعیین می‌کند. متغیر transactionSize نیز آنچنان که از نامش پیداست، اندازه تراکنش را مشخص نموده و کنترل می‌نماید.

### ۳-۲-۲ مقداردهی اولیه متغیرها

لازم است که تمامی متغیرهای تعریف شده مورد مقداردهی اولیه قرار گیرند تا حالت اولیهٔ سیستم مشخص شود. در مدل ما تمامی متغیرهای عددی به ’۰‘، متغیرهای بولی به ’false‘، و متغیرهای شمارشی به ’n‘ مقداردهی اولیه می‌شوند.

### ۳-۲-۳ انتقال حالت

متغیر rank به دو صورت می‌تواند تغییر مقدار دهد، با اجرای عملیات updateRank در هنگام پردازش تراکنش‌های ورودی و یا اجرای قانون bonus-rank. بنابراین انتقال حالت این متغیر به صورت زیر تعریف می‌شود.

```
next(rank) :=
        case
          (input = uR) : (rank + ۱) mod ۵;
          (activeRule = bR) : (rank + ۱) mod ۵;
          TRUE : rank ;
        esac;
```

چگونگی تغییر مقدار متغیر bonus نیز به صورتی مشابه مشخص می‌شود.

در قسمت بعدی از مدل، شرایط تغییر متغیرهای حالت input و transactionSize مشخص می‌شود. این بخش از چهارچوب پیشنهادی، کار مدیریت و پردازش تراکنش‌های ورودی را انجام می‌دهد. در صورتی که سیستم در فاز پردازش قوانین فعال قرار نداشته و تراکنش فعلی پایان نیافته باشد، یک عملیات داده ای از ورودی دریافت شده و اندازه تراکنش یک واحد افزایش می‌یابد. با پایان تراکنش ورودی جاری، فرآیند پردازش قوانین آغاز شده و پردازش تراکنش‌های ورودی متوقف خواهد شد.

```
next(input) :=
        case
          (ruleProcessing = FALSE) & (transactionSize > ۰) & (transactionSize < ۲) : {uR, uB};
          TRUE : n;
        esac;
next(transactionSize) :=
        case
          !(input = n) : (transactionSize + ۱) mod ۳;
         (ruleProcessing = FALSE) & (transactionSize = ۰) : transactionSize + ۱;
          TRUE : transactionSize;
        esac;
```

از متغیرهای rank-bonus و bonus-rank به منظور مدل‌سازی فرآیند تحریک قوانین استفاده می‌کنیم. در واقع این متغیرها وظیفه واحدهای تشخیص رویداد[۹] و مدیریت شرط[۱۰] را به عهده دارند. تغییر حالت قانون rank-bonus، به صورت زیر مدل سازی شده و چگونگی تغییر مقدار متغیر bonus-rank نیز به همین سبک تعریف می‌شود.

```
next(rank-bonus) :=
        case
          !(rank = oldrank) & (rank mod ۲ = ۰) : rank-bonus + ۱;
          (active = rB) & (rank-bonus > ۰) : rank-bonus - ۱;
          TRUE : rank-bonus;
        esac;
```

متغیر activeRule با انتخاب یک قانون برای اجرا، سیستم را وارد فاز پردازش قوانین می‌کند. بنابراین در هنگام پردازش تراکنش‌های ورودی، به این متغیر مقدار 'n' اختصاص یافته و تنها زمانی می‌تواند تغییر حالت دهد که فرآیند دریافت و پردازش ورودی متوقف شده باشد.

```
Next(activeRule) :=
        case
          (input = n) & !(activeRule = bR) & (bonus-rank > ۰) : bR;
          (input = n) & !(activeRule = rB) & (rank-bonus > ۰) : rB;
          (input = n) & !(bonus = old-bonus) : bR;
          (input = n) & !(rank = old-rank) & (rank mod ۲ = ۰): rB;
          TRUE : n;
        esac;
```

متغیر ruleProceesing مشخص می‌کند که آیا سیستم در حال پردازش قوانین می‌باشد یا خیر. در واقع این متغیر وظیفه مشخص نمودن فاز جاری سیستم را به عهده دارد. ruleProceesing با اتمام پردازش تراکنش ورودی، مقدار 'True' گرفته و سیستم وارد فاز پردازش قوانین می‌شود. هرگاه قانونی تحریک شده و یا تحت پردازش وجود نداشته باشد، این متغیر به 'False' تغییر مقدار داده و سیستم وارد فاز پردازش تراکنش‌های ورودی خواهد شد.

```
init(ruleProcessing) := FALSE;
    next(ruleProcessing) :=
        case
          (next(activeRule) = n) & (rank-bonus = ۰) & (bonus-rank = ۰) : FALSE;
```

---

[۹] Event Detector

[۱۰] Condition Manager

```
  (input = n) : TRUE;
  TRUE : FALSE;
esac;
```

### ۳-۳- استفاده از استراتژی‌های مختلف پردازش قوانین

به طور کلی سه نوع استراتژی ممکن است برای پردازش قوانین مورد استفاده قرار گیرد. استراتژی «بلافصل»، که قوانین به محض تحریک شدن مورد پردازش قرار می‌گیرند. استراتژی «معوق» که بر اساس آن پردازش قوانین تحریک شده، تا پایان تراکنش جاری به تعویق می‌افتد. سومین استراتژی نیز «منفصل» نام دارد که طبق آن پردازش تراکنش‌های تحریک شده، می‌تواند در هر یک از تراکنش‌های بعدی انجام گیرد.

مدلی که در قسمت قبل به توصیف چگونگی ایجاد آن پرداختیم، بر اساس استراتژی معوق عمل می‌کند که معمول‌ترین استراتژی پردازش قوانین بشمار می‌رود. به منظور استفاده از استراتژی بلافصل باید شرایطی فراهم شود تا به محض تحریک یک قانون، سیستم وارد فاز پردازش قوانین شده و تا زمان اتمام آن از دریافت ورودی اجتناب شود. با حذف شرط "input=n" از ابتدای عبارات شرطی در قسمت مربوط به تغییر حالت activeRule، حذف شرط دوم ruleProcessing و تغییر نتیجهٔ شرط پیش فرض آن به ’TRUE‘، این شرایط فراهم می‌شود. اما تحت استراتژی منفصل، دیگر نیازی به کنترل جریان تراکنش‌های ورودی به منظور پردازش قوانین، بین تراکنش‌ها و یا در میان تراکنشی خاص وجود نخواهد داشت. بنابراین شرط‌های توقف پردازش تراکنش‌های ورودی برداشته شده و فاز پردازش قوانین تنها زمانی آغاز می‌شود که ورودی خودبخود و بصورت تصادفی مقدار ’n‘ گرفته و متوقف شود.

### ۳-۴- مشخصه سازی

پس از ساخت مدل، نوبت به مشخصه سازی رسمی خصوصیت خاتمه پذیری می‌رسد. ما این خصوصیت را توسط هر دو منطق مرسوم در دنیای وارسی مدل، یعنی منطق درخت محاسباتی یا CTL و منطق موقت خطی یا LTL [۲۱]، به صورت زیر مشخصه سازی نموده‌ایم.

```
LTLSPEC  G((ruleProcessing=TRUE) -> F(ruleProcessing=FALSE))
CTLSPEC  AG((ruleProcessing=TRUE) -> AF(ruleProcessing=FALSE))
```

بدین معنی که در کل فضای حالت هرگاه ruleProcessing = TRUE بوده و سیستم از فاز پردازش ورودی وارد فاز پردازش قوانین شود، آنگاه حتماً در نهایت متغیر ruleProcessing مقدار ’FALSE‘ گرفته و این فاز خاتمه می‌یابد.

## ۴- نتایج وارسی مدل

با ایجاد مدل و مشخصه سازی خصوصیت مورد نظر، می‌توان اقدام به وارسی مدل نمود. برای این منظور لازم است که ابتدا مدل ایجاد شده را مسطح نموده[۱۱] و سپس به صورت دودویی در آوریم. NuSMV۲ برای این مراحل آماده سازی مدل، توابع و دستورات لازم را فراهم آورده است تا این کارها به صورت خودکار و سریع انجام پذیرند. خروجی NuSMV پس از وارسی مدل تحت دو منطق CTL و LTL به صورت زیر است.

```
-- specification AG(ruleProcessing = True -> AF ruleProcessing = False) is true
-- specification G(ruleProcessing = True -> F ruleProcessing = False) is true
```

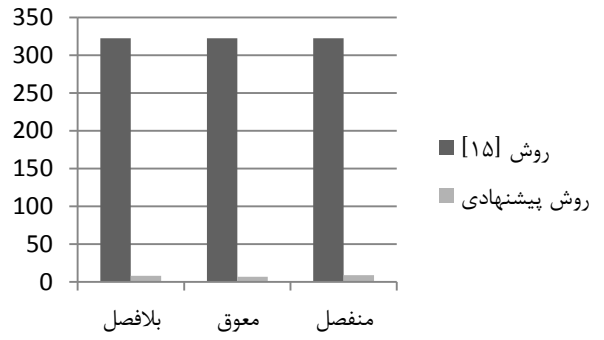
ما وارسی مدل را برای بررسی خاتمه پذیری بر روی چهارچوب پیشنهادی، تحت استراتژی‌های مختلف پردازش قوانین انجام داده‌ایم. نتایج مربوط به مدت زمان صرف شده توسط سیستم، تعداد گره های BDD اختصاص یافته، و حافظه مورد استفاده برای انجام وارسی مدل در جدول ۱ آورده شده است.

---

[۱۱] flatten

**جدول ۱ – نتایج و اطلاعات مربوط به وارسی مدل چهارچوب پیشنهادی**

| | تعداد گره های BDD | حافظه (کیلو بایت) | زمان (ثانیه) |
|---|---|---|---|
| بلافصل | ۵۴۹۸ | ۸۴۳۶ | ۰.۰۳۰ |
| معوق | ۱۶۴۵۰ | ۷۰۷۴ | ۰.۰۲۰ |
| منفصل | ۲۴۰۲۹ | ۸۹۴۲ | ۰.۰۴۰ |




روش

روش پیش

**شکل ۱ . نمودار مقایسه حافظه مورد نیاز برای وارسی مدل**

**شکل ۲. نمودار مقایسه زمان انجام وارسی مدل**

پیش‌تر اشاره نمودیم که آخرین و کامل‌ترین روش مبتنی بر وارسی مدل که پیش از این ارائه شده است، چهارچوب پیشنهادی در [۱۵] می‌باشد. در نمودارهای زیر، نتایج مربوط به روش ما با روش مذکور، تحت استراتژی‌های یکسان پردازش قوانین مورد مقایسه قرار گرفته‌اند. نمودار شکل۱، میزان حافظه مورد نیاز و نمودار شکل۲، مدت زمان صرف شده توسط سیستم را بین روش پیشنهادی ما و روش مذکور مقایسه می‌کنند.

لازم به ذکر است که نتایج آورده شده در این بخش، حاصل وارسی مدل پیشنهادی با نسخه ۲.۵.۳ از NuSMV می‌باشد که کد منبع آن از سایت رسمی این وارس مدل [۲۲] دریافت شده است. برای کامپایل نمودن NuSMV و انجام وارسی مدل، یک کامپیوتر شخصی با سیستم عاملی مبتنی بر لینوکس، پردازنده Core۲Duo ۲.۵۳GHz اینتل و ۴ گیگابایت حافظه اصلی، مورد استفاده قرار گرفته است.

## ۵- نتیجه گیری

ما در این مقاله روش و چهارچوبی جدید ارائه نمودیم برای بررسی صحت رفتاری قوانین فعال با استفاده از وارسی مدل نمادین مبتنی بر BDD و SAT. ابتدا چهارچوب پیشنهادیمان را بر اساس سیستم نمونه توصیف نموده و نشان دادیم که چگونه می‌توان مدل ورودی مورد نیاز را برای انجام وارسی مدل با NuSMV۲ ایجاد نمود. آنگاه چگونگی مشخصه سازی خصوصیت خاتمه پذیری را برای مدل ایجاد شده، تحت دو منطق LTL و CTL بیان نمودیم.

هدف ما ارائه روشی بود که در بررسی خاتمه پذیری قوانین فعال محافظه کارانه و ناقص نباشد، از استراتژی‌های مختلف پردازش قوانین پشتیبانی کند، و از همه مهم‌تر امکان اجتناب از مشکل انفجار فضای حالت را فراهم آورد. نتایج بدست آمده نشان می‌دهد که ما در تأمین اهداف خود موفق بوده‌ایم. بر خلاف اغلب روش‌های پیشنهادی در گذشته که صرفاً مبتنی بر سیستم پایگاه داده می‌باشند [۷] [۸] [۱۳] [۱۵] [۱۰]، چهارچوب پیشنهادی ما بر محور قوانین فعال بنا شده و سعی شده است که از ورود به ساختار پایگاه داده فعال و وابستگی به آن اجتناب شود. بدین ترتیب روش پیشنهادی برای دیگر سیستم‌های مبتنی بر قوانین فعال قابل تطبیق و استفاده خواهد بود.

در آینده می‌توان با افزودن قابلیت بررسی دو خصوصیت هم‌آمیزی و قطعیت شهودی، این روش را کامل‌تر نمود. از دیگر کارهایی که می‌تواند در این راستا انجام گیرد، توسعه ابزاری ست با قابلیت اتصال به NuSMV، که قادر باشد با دریافت مشخصات سیستم و مجموعهٔ قوانین، مدل را به صورت خودکار ایجاد و وارسی نماید.

## مراجع

[۱] M. Zoumboulakis, G. Roussos, and A. Poulovassilis, "Active rules for sensor databases," in *the ۱st international workshop on Data management for sensor networks: in conjunction with VLDB ۲۰۰۴*, New York, USA, ۲۰۰۴.

[۲] Jie Bao, Lalana Kagal, Ian Jacobi, Li Ding and James Hendler Ankesh Khandelwal, "Analyzing the AIR Language: A Semantic Web (Production) Rule Language ," *Web Reasoning and Rule Systems*, vol. ۶۳۳۳/۲۰۱۰, pp. ۵۸-۷۲, ۲۰۱۰.

[۳] A Bonifati and S Ceri, "Active rules for XML: A new paradigm for E-services," *The VLDB Journal*, pp. ۳۹–۴۷, ۲۰۰۱.

[۴] SM Huang, J Tait, and CH Su, "Using Active Rules to Maintain Data Consistency in Data Warehouse Systems," *Progressive Methods in Data Warehousing and Business Intelligence: Concepts and Competitive Analytics*, pp. ۲۵۲-۲۷۲, ۲۰۰۹.

[۵] Josef Schiefer, Szabolcs Rozsnyai, Christian Rauscher, and Gerd Saurer, "Event-driven rules for sensing and responding to business situations," in *the ۲۰۰۷ inaugural international conference on Distributed event-based systems* , New York,USA, ۲۰۰۷.

[۶] N.W. Paton and O. Diaz, "Active database systems," *ACM Computing Surveys (CSUR)*, vol. ۳۱ , no. ۱, March ۱۹۹۹.

[۷] J.A. Bailey, G. Dong, and K. Ramamohanarao, "Decidability and undecidability results for the termination problem of active database rules," *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pp. ۲۶۴-۲۷۳, ۱۹۹۸.

[۸] A. Aiken, J. Hellerstein, and J. Widom, "Static analysis techniques for predicting the behavior of active database rules," *ACM Transactions on Database Systems*, pp. ۳-۴۱, ۱۹۹۵.

[۹] D. Zimmer, A. Meckenstock, and R. Unland, "Using petri nets for rule termination analysis," no. Proceedings of the workshop on Databases, pp. ۲۹-۳۲, ۱۹۹۷.

[۱۰] E. Baralis and J. Widom, "An algebraic approach to static analysis of active database rules," *ACM Transactions on Database Systems*, pp. ۲۶۹-۳۳۲, ۲۰۰۰.

[۱۱] Edmund M. Clarke, The Birth of Model Checking , ۵۰۰۰ ۲۰۰۸.

[۱۲] Christel Baier and Joost-Pieter Katoen, *Principles of Model Checking*.: the MIT Press, ۲۰۰۸.

[۱۳] Tarek Ghazi, Tarek S. Ghazi, and Michael Huth, "A framework for model checking active database management systems," Department of computing and information sciences, Kensas State University, Manhattan, ۱۹۹۸.

[۱۴] Ray Indrakshi and Ray Indrajit, "Detecting Termination of Active Database Rules Using Symbolic Model Checking," *Advances in Databases and Information Systems, Lecture Notes in Computer Science*, vol. ۲۱۵۱, pp. ۲۶۶-۲۷۹, ۲۰۰۱.

[۱۵] T Tsuchiya, C Eun-Hye, and T Kikuno, "Model Checking active database rules," *the IPSJ Transactions on Databases*, vol. ۴۷, no. SIG۱۹(TOD۳۲), pp. ۱۴-۲۷, ۲۰۰۶.

[۱۶] GJ Holzmann, "The model checker SPIN," *IEEE Transactions on Software Engineering*, ۱۹۹۷.

[۱۷] A. Cimatti, E.M. Clarke, E. Giunchiglia, and F. Giunchiglia, "NuSMV ۲: An OpenSource Tool for Symbolic Model Checking," in *the ۱۴th International Conference on Computer Aided Verification*, London, ۲۰۰۲, pp. ۳۵۹-۳۶۴.

[۱۸] KL McMillan, *Symbolic Model Checking*.: Kluwer Academic Publisher, ۱۹۹۳.

[۱۹] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik, "Chaff: Engineering an Efficient SAT Solver," in *۳۹th Design Automation Conference (DAC)*, Las Vegas, ۲۰۰۱.

[۲۰] Niklas Een and Niklas Sörensson, "MiniSat - A SAT Solver with Conflict-Clause Minimization," in *SAT ۲۰۰۵*, ۲۰۰۵.

[۲۱] E. A. Emerson, "Temporal and modal logic," in HANDBOOK OF THEORETICAL COMPUTER SCIENCE.: Elsevier, ۱۹۹۵, pp. ۹۹۵-۱۰۷۲.

[۲۲] NuSMV: a new symbolic model checker. [Online]. http://nusmv.fbk.eu/