مقاله پژوهشی

تولید خودکار آزمایه مبتنی بر توصیف رفتاری قاعده‌محور

علی حبیبی و رامتین خسروی

چکیده: با رشد روزافزون استفاده از نرم‌افزارها در کاربردهای ایمنی- بحرانی نظیر صنعت خودرو، صنایع دفاعی و صنایع پزشکی، کسب سطوح بالای اطمینان از کیفیت این نرم‌افزارها امری ضروری است. آزمون مبتنی بر مدل به عنوان یک روش تولید خودکار آزمایه از طرفی با پوشش‌دادن یک توصیف صوری از کارکرد سامانه اطمینانی نسبی ایجاد می‌کند که سناریوهای مختلف اجرای برنامه آزموده می‌شوند و از طرف دیگر با خودکارسازی تولید این آزمایه‌ها هزینه تولید آزمون را به شکل چشم‌گیری کاهش می‌دهد. در این پژوهش یک چارچوب آزمون مبتنی بر مدل ارائه شده که از یک مدل قاعده‌محور استفاده می‌کند و بر اساس دو معیار پوشش قاعده و پوشش شرط فعال قاعده توانایی تولید آزمایه دارد. برای تولید آزمایه، این چارچوب از یک رویکرد جستجومحور مبتنی بر الگوریتم ژنتیک استفاده می‌کند. روش پیشنهادی امکان تعریف یک سامانه با فضای حالت بزرگ و تولید آزمایه از آن را ارائه می‌دهد. این چارچوب با انجام مطالعه موردی روی یک نرم‌افزار نهفته صنعتی ارزیابی شده و نتایج ارزیابی‌ها نشان از کاربردی‌بودن آن در مسائل واقعی در صنعت دارند.

***کلیدواژه:* آزمون مبتنی بر مدل، آزمون جستجومحور، توصیف صوری.**

۱- مقدمه

امروزه استفاده از نرم‌افزارهای ایمنی- بحرانی[1] در صنایع مختلف بسیار گسترش یافته است. به عنوان نمونه در سال‌های اخیر، همراه با جایگزینی فناوری‌های مکانیکی و هیدرولیکی با فناوری‌های الکترونیکی در صنعت خودرو، نقش نرم‌افزار بسیار پررنگ شده است [۱]؛ به طوری که بیش از ۹۰٪ نوآوری‌های جاری در صنعت خودرو در بخش‌های الکترونیکی و نرم‌افزاری است و با اضافه‌شدن فناوری‌ها و ویژگی‌های نوظهور به خودروها، نرم‌افزار موجود در یک خودروی نوین از مرز صد میلیون خط کد عبور کرده است [۲]. از سوی دیگر مأموریت حساس سامانه‌های نهفته در این گونه صنایع نیاز به اطمینان از درستی عملکرد آنها را بسیار حیاتی می‌کند. بنابراین داشتن یک روش سازمان‌یافته آزمون نرم‌افزار که بتواند در سطوح مختلف، کارکرد سیستم را بر مبنای نیازمندی‌ها یا مشخصات طراحی بسنجد، اهمیت زیادی دارد [۳]. تولید خودکار آزمایه[2] از فنونی است که می‌توان به این منظور به کار گرفت. از بین روش‌های مختلف تولید خودکار آزمایه، آزمون مبتنی بر مدل[3] به خاطر نقش‌آفرینی مدل سیستم به عنوان پیش‌گوی آزمون[4] یکی از اثربخش‌ترین روش‌ها به شمار می‌آید [۴].

تاکنون روش‌ها و ابزارهای زیادی در حوزه آزمون مبتنی بر مدل ارائه شده‌اند [۵] که در دامنه‌های مختلفی از جمله نرم‌افزارهای نهفته [۶] و [۷]، سامانه‌های وب [۸] و سامانه‌های پردازش تراکنش‌های مالی [۹] و [۱۰] به کار رفته‌اند. روش‌های مطرح برای آزمون مبتنی بر مدل سامانه‌های نهفته، تولید آزمایه را بر پایه یک مدل سطح پایین انجام می‌دهند که حالات سامانه را به صورت صریح نمایش می‌دهد [۱۱] تا [۱۵]. با گسترش استفاده از آزمون مبتنی بر مدل در صنعت و همین طور پیچیده‌ترشدن سامانه‌های نهفته، بیش از پیش نیاز به بهبود مقیاس‌پذیری در این حوزه حس می‌شود [۱۶] و [۱۷]؛ زیرا در توصیف یک سامانه واقعی که با تعداد کلانی از حالات و گذارها روبه‌رو هستیم، تعریف سطح پایین و تولید آزمون از آن به‌صرفه و در مواردی امکان‌پذیر نیست. در مواردی نیز تلاش شده که آزمون مبتنی بر مدل با روش‌های دیگری ترکیب شود تا کارایی آن افزایش پیدا کند که می‌توان به مواردی مانند روش‌های تکاملی [۱۸] یا یادگیری ماشین [۱۹] اشاره کرد.

برای حل این چالش، چارچوب‌هایی برای آزمون مبتنی بر مدل ارائه شده‌اند که از توصیف سطح بالا استفاده می‌کنند؛ اما اغلب این چارچوب‌ها نیز در فرایند تولید آزمایه، توصیف مدل ابتدایی را به یک مدل سطح پایین ترجمه می‌کنند و سپس با استفاده از مدل سطح پایین تولیدشده و با روش‌های مرسوم ساختاری، اقدام به تولید آزمایه می‌کنند.

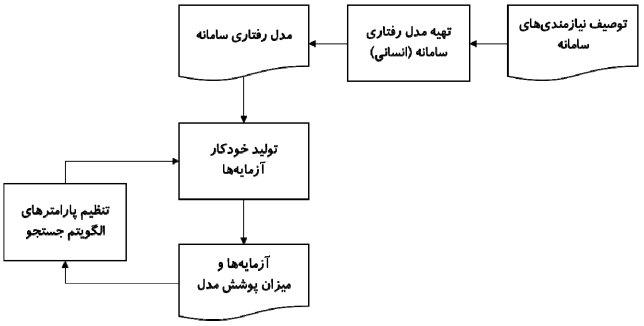
یکی از شناخته‌شده‌ترین ابزارهایی که بر این مبنا تولید شده است، fMBT می‌باشد که امکان مدل‌سازی و تولید آزمایه را ارائه می‌دهد [۲۰]. مدل استفاده‌شده در این ابزار، زبان سطح بالای پیش- پس‌شرطی AAL/Python است که ترکیبی از یک زبان ابداعی و زبان برنامه‌نویسی پایتون می‌باشد. این زبان از تعدادی قاعده و متغیر تشکیل می‌شود. هر قاعده یک شرط و یک کنش دارد که در بخش شرط، متغیرهای سازنده مدل ارزیابی می‌شوند و در بخش کنش با استفاده از زبان پایتون، مقدار متغیرها ویرایش می‌شوند. برای تولید آزمایه این قواعد به صورت یک گراف جهت‌دار بازنویسی می‌شوند؛ به این ترتیب که در هر گره به ازای تمامی قواعدی که شرط صحیح دارند، یال خروجی وجود دارد. گره مقصد یال‌های خروجی بر اساس کنش قاعده متناظر آنها تعیین می‌شود؛ بنابراین این ابزار از یک مدل سطح بالا استفاده می‌کند، ولی برای تولید آزمایه این مدل به یک مدل سطح پایین تبدیل می‌شود. رویکرد انتخاب آزمایه نیز بر اساس پوشش‌های ساختاری گراف است که از الگوریتم‌های حریصانه برای دستیابی به آن کمک گرفته می‌شود. پس اگر تعداد قواعد یا تعداد متغیرها افزایش یابد، این ابزار برای تولید مدل سطح پایین و یا ایجاد آزمایه با پوشش بالا با مشکل مواجه خواهد شد. باید توجه داشت که مدل

این مقاله در تاریخ ۲۲ شهریور ماه ۱۴۰۲ دریافت و در تاریخ ۲۸ بهمن ماه ۱۴۰۲ بازنگری شد.

علی حبیبی، دانشکده مهندسی برق و کامپیوتر، دانشکدگان فنی، دانشگاه تهران، تهران، ایران، (email: habibi.ali@ut.ac.ir).

رامتین خسروی (نویسنده مسئول)، دانشکده مهندسی برق و کامپیوتر، دانشکدگان فنی، دانشگاه تهران، تهران، ایران، (email: r.khosravi@ut.ac.ir).

1. Safety-Critical
2. Automatic Test Case Generation
3. Model-Based Testing
4. Test Oracle



شکل ۱: فرایند کلی استفاده از روش پیشنهادی برای آزمون مبتنی بر مدل.

میانی به‌دست‌آمده از مدل‌های سطح بالای واقعی در صنعت می‌تواند گاه تا میلیون‌ها حالت و گذار داشته باشد که برای به‌کارگیری این رویکرد چالشی جدی محسوب می‌شود. تحقیقات و ابزارهای دیگری نیز در این رده وجود دارند که از بین آنها می‌توان به [۲۱] و [۲۲] اشاره کرد که آنها نیز در مدل‌سازی سامانه‌های بزرگ صنعتی با چالش مشابهی مواجه خواهند بود.

رویکردهای دیگری نیز برای آزمون مبتنی بر مدل معرفی شده‌اند که بدون ترجمه مدل سطح بالا به مدل میانی، اقدام به تولید آزمایه می‌کنند. اما نقطه ضعفی که در این روش‌ها مشاهده می‌شود، نبود معیار مناسبی برای ارزیابی کیفیت مجموعه آزمون تولیدشده است. به عنوان مثال با روش‌هایی مانند الگوریتم انتخاب تصادفی برای تولید آزمون اقدام می‌کنند که ارزیابی مناسبی از کیفیت مجموعه آزمون ارائه نمی‌دهد. یکی از مهم‌ترین ابزارهایی که این روش را دنبال می‌کند، تورکساکیس[1] است که بر اساس نظریه تطابق ورودی- خروجی (ioco)[2] بنا شده است [۲۳]. به دو صورت می‌توان یک مدل تورکساکیس را توصیف کرد: یکی تعریف سطح پایین و مستقیم سامانه گذار ورودی- خروجی و دیگری با استفاده از جبر پردازه[3]. در روش اول با یک زبان متنی، توصیف به صورت یک سامانه گذار ورودی- خروجی نوشته می‌شود؛ اما در روش دوم با استفاده از جبر پردازه، یک تعریف سطح بالاتر از سامانه ارائه می‌شود که حالات و گذارهای سامانه را به صورت مستقیم تعریف نمی‌کند. شیوه انتخاب آزمایه در تورکساکیس به صورت تصادفی است و در حال حاضر معیاری برای بررسی کیفیت آزمایه‌های تولیدی آن معرفی نشده است. تحقیقات دیگری نیز در این زمینه انجام شده است که از بین آنها می‌توان به [۲۴] و [۲۵] اشاره کرد.

در این پژوهش روشی ارائه کردیم تا بتوان سامانه‌های بزرگ صنعتی را با یک زبان سطح بالا مدل‌سازی کرد و تولید خودکار آزمایه را بر اساس پوشش مدل مذکور هدایت نمود. مدل ارائه‌شده رفتار سامانه تحت آزمون را مبتنی بر رخدادها و قواعد توصیف می‌کند و به منظور پشتیبانی از سامانه‌های بزرگ صنعتی، مدل سامانه را در قالب تعدادی مؤلفه بیان می‌کند. این پژوهش از روش‌های انتخاب آزمون مبتنی بر مدل بهره برده است و معیار پوشش قاعده و شرط فعال قاعده به عنوان معیارهای ارزیابی کیفیت آزمون معرفی شده‌اند. تولید خودکار آزمایه نیز با الگوی جستجومحور و با بهره‌گیری از الگوریتم ژنتیک با هدف بهینه‌سازی پوشش مدل و کم‌کردن تعداد آزمایه‌ها صورت می‌پذیرد. شکل ۱ نمایی کلی از فرایند استفاده از روش پیشنهادی را نمایش می‌دهد.

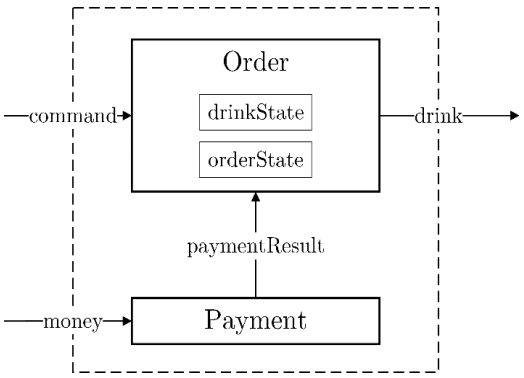
ارزیابی این روش با به‌کارگیری آن روی یک نرم‌افزار نهفته واقعی در حوزه الکترونیک خودرو انجام شده است. از آنجا که در صنایعی مانند خودروسازی، توصیف و مدل‌سازی رفتار نرم‌افزارها فعالیتی پذیرفته‌شده است و به عنوان جزء مهمی از فرایندهای تولید نرم‌افزار به آن عمل می‌شود، تولید مدل به طور خودکار از روی توصیف‌های موجود صورت گرفته و عملاً با هزینه‌ای ناچیز امکان تولید خودکار آزمایه را فراهم کرده است.

۲- توصیف کارکرد سامانه

برای به‌کارگیری روش آزمون مبتنی بر مدل، لازم است مدل رفتاری سامانه در یک زبان متناسب با کاربرد و با نحو و معناشناسی صوری توصیف گردد. زبان‌های مدل‌سازی مختلفی می‌توانند به این منظور مورد استفاده قرار بگیرند؛ نظیر پرومولا[4] [۲۶]، ربکا[5] [۲۷]، ایونت- بی[6] [۲۸] یا زبان‌های مبتنی بر جبر پردازه [۲۹]. در انتخاب زبان مدل‌سازی، علاوه بر تناسب زبان با ویژگی‌های دامنه مسئله، لازم است به یک نکته در مورد محیط به‌کارگیری آن توجه داشت. با توجه به این که در پژوهش حاضر وجه قابلیت استفاده در صنعت مورد تأکید بوده است، نزدیک‌بودن ساختار زبان مدل‌سازی به مدل‌های موجود در صنعت باعث کم‌شدن هزینه تولید مدل‌های صوری، کم‌شدن احتمال خطا در تبدیل مدل‌ها و نهایتاً منجر به تسهیل پذیرش روش پیشنهادی از طرف صنعت می‌شود. همان گونه که پیش از این اشاره شد، این پژوهش در راستای آزمودن سامانه‌های مورد استفاده در شرکت کروز صورت گرفته است. ساختار مدل‌های رفتاری مورد استفاده در این شرکت به صورت جداولی است که هر سطر آن یک «قاعده» را توصیف می‌کند که بر اساس مقادیر ورودی‌ها و متغیرهای حالت، تغییراتی در خروجی‌ها و متغیرهای حالت ایجاد می‌نماید. به دلیل عدم وجود یک توصیف صوری[7] از این روش مدل‌سازی، ابهام‌ها و تناقضات متعددی در این مدل‌ها به چشم می‌خورد. برای رفع این مشکل، یک زبان مدل‌سازی با ساختاری نزدیک به مدل‌های شرکت طراحی گردید و مدل‌های مذکور پس از رفع ابهام‌ها و تناقضات در این زبان توصیف شدند. طبیعتاً این کار می‌توانست با استفاده از زبان‌های مدل‌سازی موجود انجام شود؛ اما در بیشتر موارد توان صرف‌شده برای تبدیل مدل‌های شرکت به مدل‌های صوری به طور قابل ملاحظه‌ای افزایش می‌یافت.

زبان مدل‌سازی پیشنهادی از یک مدل مؤلفه‌ای قاعده‌محور[8] بهره می‌برد؛ به این معنا که سامانه با توصیف چند مؤلفه و ترکیب آنها تعریف می‌شود. هر مؤلفه از تعدادی قاعده تشکیل گردیده و هر قاعده خود بر کوچک‌ترین عناصر سازنده این مدل یعنی متغیرها بنا شده است. قاعده، تکه اطلاعاتی از نیازمندی‌ها را به صورت صوری و ساختارمند نشان می‌دهد. قواعد موجود در این مدل، مبتنی بر رخداد و حالت هستند؛ به این معنا که هر قاعده از طرفی دارای شرط‌هایی است که مبتنی بر حالت سامانه (یا به عبارت دقیق‌تر مؤلفه) هستند و از سوی دیگر تنها در صورت فعال‌شدن رخداد خود قابل اجرا خواهد بود. به عبارت دیگر در صورتی که شرط‌های یک قاعده صحیح باشند و رخداد تعریف‌شده آن فعال شود، می‌گوییم که قاعده قابل اجراست. حاصل اجرای یک قاعده در بخش

1. TorXakis
2. Input-Output Conformance
3. Process Algebra
4. Promela
5. Rebeca
6. Event-B
7. Formal Specification
8. Rule-Based



شکل ۲: نمودار بلوکی دستگاه فروش خودکار.

کنش آن نوشته می‌شود. کنش یک قاعده از یک مجموعه انتساب به متغیرهای داخلی یا خروجی آن تشکیل می‌شود.

اگرچه تمرکز این پژوهش بر مدل‌های بزرگ‌تر است، اما در این فصل برای سهولت در نمایش با استفاده از یک مثال بسیار ساده، تعاریف و توضیحات هر بخش را بیان می‌کنیم. سامانه نمونه، یک دستگاه فروش خودکار است که در ازای دریافت هزینه، چای یا قهوه تحویل می‌دهد. توصیف این دستگاه از دو مؤلفه تشکیل شده که در یکی از آنها سفارش گرفته می‌شود و دیگری وظیفه دریافت پول را بر عهده دارد. شکل ۲ نمودار بلوکی این سامانه را نشان می‌دهد. این دستگاه توسط دو مؤلفه توصیف شده که هر کدام یک ورودی از محیط دریافت می‌کنند. متغیر paymentResult بین این دو مؤلفه ارتباط برقرار می‌کند و تنها خروجی سامانه از مؤلفه Order است. دو متغیر داخلی در این سامانه تعریف شده که هر دو در مؤلفه Order قرار دارند.

۲-۱ نحو توصیف مدل

به علت ماهیت فرایند آزمون و نقش‌آفرینی گروه‌های مختلف اعم از فنی و غیرفنی، نیاز است تا نحو پیشنهادی برای تمامی ذی‌نفعان این فرایند قابل درک باشد. نحو پیشنهادی با الهام از EARS [۳۰] طراحی شده و در شکل ۳ آمده است. در نحو پیشنهادی سه بخش کلی داریم و تعریف متغیرها با کلمه کلیدی VARS مشخص می‌شود. در این بخش متغیرهای داخلی پس از کلمه کلیدی CMPINT و متغیرهای خارجی پس از کلمه کلیدی CMPEXT تعریف می‌شوند. در اینجا دامنه هر متغیر مشخص می‌شود و مقدار اولیه آن بین دو قلاب قرار می‌گیرد. در بخش دوم تعریف مؤلفه‌ها با کلمه کلیدی COMP مشخص می‌شود و هر مؤلفه را جداگانه می‌توان تعریف کرد. برای تعریف یک مؤلفه باید متغیرهای ورودی، متغیرهای خروجی، متغیرهای داخلی و قاعده‌های آن تعریف شوند. در یک قاعده، رخداد پس از کلمه کلیدی WHEN، شرط‌ها پس از کلمه کلیدی WHILE و کنش‌ها پس از کلمه کلیدی THEN تعریف می‌شوند و نهایتاً تعریف سامانه با کلمه کلیدی SYSTEM مشخص می‌شود. در این چارچوب می‌توان با یک مجموعه از متغیرها و ترکیب مؤلفه‌های تعریف‌شده، سامانه‌های متفاوتی را توصیف کرد. به عنوان نمونه در شکل ۴، دستگاه فروش خودکار در قالب دو مؤلفه مدل شده است.

در این مدل، ابتدا ۴ متغیر cmd، drink، money و paymentResult به عنوان متغیرهای خارجی تعریف شده‌اند. مجموعه مقادیر مجازی که هر یک از این متغیرها می‌توانند اختیار کنند نیز در این بخش مشخص شده است. مقدار اولیه هر متغیر نیز در علامت‌های «[» و «]» ذکر شده‌اند. پس از آن، دو متغیر داخلی به نام‌های orderState و drinkState تعریف شده‌اند. بعد از تعریف متغیرها، مؤلفه‌های Order و Payment توصیف شده‌اند. در تعریف هر یک از مؤلفه‌ها ورودی‌ها، خروجی‌ها و متغیرهای داخلی آن مؤلفه ذکر شده‌اند. پس از آن، رفتار مؤلفه در قالب چند قاعده توصیف می‌شود. مثلاً قاعده selectCoffee بیان می‌کند که هر زمان مقدار متغیر cmd برابر coffee شد (بخش WHEN)، در صورتی که مقدار orderState برابر notSelected باشد (بخش WHILE)، آن گاه بر اثر اجرای این قاعده، مقدار دو متغیر orderState و drinkState به مقادیر مشخص‌شده تغییر خواهد کرد.

۲-۲ معناشناسی مدل

در این بخش معناشناسی صوری مدل پیشنهادی را در قالب معناشناسی عملیاتی ارائه خواهیم کرد. در ابتدا معنای ریاضی ساختار مدل را توصیف می‌کنیم و بعد از آن معناشناسی پویای آن را در قالب یک سیستم گذار مشخص خواهیم کرد. در توصیف معناشناسی مدل تا حد امکان از نمادگذاری‌های استاندارد استفاده کرده‌ایم. برخی نمادهایی که کمتر شناخته شده‌اند را در این بخش مرور می‌کنیم. اگر مجموعه K را داشته باشیم، مجموعه K^* مجموعه‌ای از تمامی دنباله‌های متناهی روی عناصر موجود در K است. برای یک دنباله $k \in K^*$ با طول n، نماد k_i نمایانگر عنصر iام دنباله است $(1 \le i \le n)$. در این شرایط k را به صورت $\langle k_1, k_2, \ldots, k_n \rangle$ می‌نویسیم. دنباله تهی را با ϵ نمایش می‌دهیم و $\langle h|T \rangle$ دنباله‌ای را بیان می‌کند که عنصر اول آن $h \in K$ است و $T \in K^*$ دنباله شامل بقیه عناصر k را نشان می‌دهد. برای دو دنباله α و α' روی K، $\alpha \oplus \alpha'$ یک دنباله از اضافه‌کردن α' به انتهای α است. برای تابع $f : X \to Y$ از نشانه‌گذاری $f[x \mapsto y]$ برای نمایش تابع $\{(a,b) \in f \,|\, a \neq x\} \cup \{(x,y)\}$ استفاده می‌کنیم و همچنین از نشانه‌گذاری $x \mapsto y$ به عنوان جایگزینی برای (x,y) استفاده می‌شود.

۲-۲-۱ ساختار مدل

برای تعریف صوری معناشناسی مدل پیشنهادی، ابتدا لازم است تا ساختار آن را ارائه کنیم. یک مدل پیشنهادی از یک بخش متغیرها، یک یا چند مؤلفه و یک یا چند سامانه تشکیل می‌شود که در فرایند آزمون می‌توان برای این سامانه‌ها به صورت مجزا تولید آزمایه کرد.

مجموعه متغیرها در مدل پیشنهادی $Var = CmpInt \cup CmpExt$ می‌باشد که $CmpInt \cap CmpExt = \varnothing$ است. این مجموعه از ۲ مجموعه مجزای متغیرهای داخلی $CmpInt$ و متغیرهای خارجی $CmpExt$ تشکیل شده است. یک متغیر داخلی حالت یک مؤلفه را نشان می‌دهد و یک متغیر خارجی ارتباط بین یک مؤلفه با یک یا چند مؤلفه، ورودی و یا خروجی سامانه را نشان می‌دهد. دامنه متغیر $x \in Var$ را با $dom(x)$ نمایش می‌دهیم.

یک نمونه از نوع $Comp = CName \times 2^{Var} \times 2^{Var} \times 2^{Var} \times Rule^*$ هر مؤلفه را در نظر می‌گیریم که در آن $CName$ مجموعه تمامی نام‌های مؤلفه‌های مدل، Var مجموعه تمامی متغیرهای مدل و $Rule$ مجموعه تمامی قاعده‌های مدل را نشان می‌دهند. در اینجا $(cname, inp, int, otp, r)$ مؤلفه‌ای با نام $cname$، مجموعه ناتهی متغیرهای ورودی $inp \subset CmpExt$، مجموعه متغیرهای داخلی $int \subseteq CmpInt$، مجموعه ناتهی متغیرهای خروجی $otp \subset CmpExt$ و دنباله ناتهی قواعد r را داراست. در صورتی که $inp \cap otp \neq \varnothing$ باشد، مؤلفه دارای متغیرهای بازخورد خواهد بود.

یک نمونه از نوع $Rule = RName \times 2^{Clause} \times Event \times Action$ هر قاعده را در نظر می‌گیریم که در آن $RName$ مجموعه تمامی نام‌های قاعده‌ها، $Clause$ مجموعه تمامی بندها روی متغیرهای داخلی و ورودی

⟨*specification*⟩ ::= ⟨*vars*⟩ ⟨*components*⟩$^+$ ⟨*system*⟩$^+$
⟨*vars*⟩ ::= **"VARS" "{" "CMPEXT"** ⟨*initial*⟩$^+$ **"CMPINT"** ⟨*initial*⟩* **"}"**
⟨*initial*⟩ ::= ⟨*varName*⟩ **"="** **"("** (⟨*dmList*⟩ | ⟨*dmRange*⟩) **")"**
⟨*dmList*⟩ ::= **"["** ⟨*value*⟩ **"]"** (**","** ⟨*value*⟩)*
⟨*dmRange*⟩ ::= **"["** ⟨*Int*⟩ **"]"** **","** ⟨*Int*⟩ **":"** ⟨*Int*⟩
⟨*components*⟩ ::= **"COMP"** ⟨*compName*⟩ **"{"** ⟨*inputs*⟩ ⟨*internals*⟩ ⟨*outputs*⟩ ⟨*rules*⟩ **"}"**
⟨*inputs*⟩ ::= **"INPUTS"** ⟨*varsList*⟩
⟨*internals*⟩ ::= **"INTERNALS"** (⟨*varsList*⟩ | ϵ)
⟨*outputs*⟩ ::= **"OUTPUTS"** ⟨*varsList*⟩
⟨*rules*⟩ ::= **"RULES"** ⟨*rule*⟩$^+$
⟨*rule*⟩ ::= ⟨*ruleName*⟩ **"{" "WHEN"** ⟨*condition*⟩ **"WHILE"** ⟨*condition*⟩* **"THEN"** (⟨*assign*⟩* | **"RESET"**) **"}"**
⟨*system*⟩ ::= **"SYSTEM"** ⟨*sysName*⟩ **"{"** ⟨*compName*⟩ (**","** ⟨*compName*⟩)* **"}"**
⟨*condition*⟩ ::= ⟨*varName*⟩ ⟨*comparisonOp*⟩ (⟨*value*⟩ | ⟨*varName*⟩)
⟨*assign*⟩ ::= ⟨*varName*⟩ **"="** (⟨*value*⟩ | ⟨*varName*⟩)
⟨*varsList*⟩ ::= ⟨*varName*⟩ (**","** ⟨*varName*⟩)*
⟨*varName*⟩ ::= ⟨*identifier*⟩
⟨*sysName*⟩ ::= ⟨*identifier*⟩
⟨*ruleName*⟩ ::= ⟨*identifier*⟩
⟨*compName*⟩ ::= ⟨*identifier*⟩
⟨*comparisonOp*⟩ ::= **"=="** | **">"** | **"<"** | **">="** | **"<="**
⟨*value*⟩ ::= ? Boolean, Integer, and String literals ?

شکل ۳: دستور زبان مستقل از متن مدل پیشنهادی.

```
VARS {
CMPEXT
  cmd = ([clear], coffee, tea, sugar, produce, take)
  drink = ([none], coffee, tea, coffeeWithSugar)
  money = ([false], true)
  paymentResult = ([false], true)
CMPINT
  orderState = ([notSelected], selected, produced)
  drinkState = ([none], coffee, tea, coffeeWithSugar)
}

COMP Order {
  INPUTS order, paymentResult
  INTERNALS orderState, drinkState
  OUTPUTS drink

  RULES
    selectCoffee {
      WHEN cmd==coffee
      WHILE orderState==notSelected
      THEN orderState=selected, drinkState=coffee
    }
    selectTea {
      WHEN cmd==tea
      WHILE orderState==notSelected
      THEN orderState=selected, drinkState=tea
    }
    addSugar {
      WHEN cmd==sugar
      WHILE orderState==selected, drinkState==coffee
      THEN drinkState=coffeeWithSugar
    }
    produce {
      WHEN cmd==produce
      WHILE orderState==selected,
            paymentResult==true
      THEN drink=drinkState, orderState=produced
    }
    take {
      WHEN cmd==take
      WHILE orderState==produced
      THEN RESET
    }
    clear {
      WHEN cmd==clear
      WHILE
      THEN RESET
    }
}

COMP Payment {
  INPUTS money
  INTERNALS
  OUTPUTS paymentResult

  RULES
    pay {
      WHEN money==true
      WHILE
      THEN paymentResult=true
    }
}

SYSTEM VendingMachine {Order, Payment}
```

شکل ۴: متن مدل دستگاه فروش خودکار.

مؤلفه، *Event* مجموعه تمامی رخدادهای قاعده‌های مدل که خود از نوع *Clause* است و *Action* مجموعه تمامی دنباله‌های کنش‌ها بر روی متغیرهای داخلی و خارجی را نشان می‌دهند. یک قاعده $(rname, c, e, a)$ در مؤلفه $(cname, inp, int, otp, r)$، دارای نام *rname*، مجموعه شروط c، رخداد e و کنش a است. اگر $c = \varnothing$، قاعده تماماً مبتنی بر رخداد خواهد بود و اگر $a = \varnothing$، قاعده عدم تغییر در سامانه را نشان می‌دهد. *Clause* مجموعه بندها روی $inp \cup int$ را تعریف می‌کند. به عبارت دیگر مجموعه‌ای از فرمول‌های منطق گزاره‌ای است که نمادهای گزاره‌ای آنها متغیرهای مجموعه $inp \cup int$ هستند. کنش‌ها از نوع $Action = Assign^* \cup \langle \text{RESET} \rangle$ هستند که $Assign = Var \times Expr$ مجموعه انتساب‌ها به متغیرهای مجموعه $int \cup otp$ را نشان می‌دهد و کلمه کلیدی RESET به معنای بازنشانی مقادیر اولیه متغیرهاست.

هر سامانه را یک نمونه از نوع $System = SName \times Comp^*$ در نظر می‌گیریم که *SName* مجموعه تمامی نام‌های سامانه‌ها و *Comp* مجموعه تمامی مؤلفه‌های تعریف‌شده در مدل را نشان می‌دهند. یک سامانه $(sname, comp)$ دارای نام *sname* و دنباله مؤلفه‌های *comp* است که $|comp| > 0$ می‌باشد.

کنش انتساب

$$\frac{y \in sinp \wedge s(x) = (v[y \mapsto u], \epsilon)}{s \to s[x \mapsto (v[y \mapsto u'], \langle \{r.e \,|\, r \in R \wedge (v[y \mapsto u] \not\models r.e) \wedge (v[y \mapsto u'] \models r.e)\} \rangle)]} \quad (۱)$$

کنش انتساب

$$\frac{r \in R \wedge s(x) = (v, \langle h \,|\, E \rangle) \wedge r.e \in h \wedge v \models (\wedge_{j=1}^{|r.c|} c_j \wedge r.e) \wedge r.a \neq \text{RESET}}{s \to s[x \mapsto (\text{effect}_v(r.a), \langle (h \setminus r.e) \,|\, E \oplus \text{trig}_v(r.a) \rangle)]} \quad (۲)$$

کنش بازنشانی

$$\frac{r \in R \wedge s(x) = (v, \langle h \,|\, E \rangle) \wedge r.e \in h \wedge v \models (\wedge_{j=1}^{|r.c|} c_j \wedge r.e) \wedge r.a = \text{RESET}}{s \to s[x \mapsto (v_0, \epsilon)]} \quad (۳)$$

شرط یا رخداد نادرست

$$\frac{r \in R \wedge s(x) = (v, \langle h \,|\, E \rangle) \wedge r.e \in h \wedge v \not\models (\wedge_{j=1}^{|r.c|} c_j \wedge r.e)}{s \to s[x \mapsto (v, \langle (h \setminus r.e) \,|\, E \rangle)]} \quad (۴)$$

رخدادهای تهی

$$\frac{s(x) = (v, \langle h \,|\, E \rangle) \wedge |h| = 0}{s \to s[x \mapsto (v, \langle E \rangle)]} \quad (۵)$$

رفتار توصیف‌شده این مدل با اجرای ترتیبی قاعده‌ها بیان می‌شود. سامانه‌های نهفته مورد بحث، بلافاصله پس از دریافت یک ورودی امکان دریافت ورودی بعدی را دارند، اما آزمون تمامی این رفتارها امکان‌پذیر نیست؛ در نتیجه باید قوانینی برای زمان واردکردن ورودی در طراحی و اجرای آزمون وضع شود. در مدل پیشنهادی، فرض بر آن است که در هر لحظه، تنها یکی از متغیرهای ورودی سامانه توصیف امکان دریافت مقدار دارد. پس از دریافت یک ورودی تا زمانی که هیچ قاعده‌ای قابل اجرا نباشد، سامانه ورودی دیگری دریافت نمی‌کند. در هر توصیف، چند سامانه قابل تعریف است. برای تولید آزمون باید سامانه هدف در توصیف مشخص شود.

۲-۲-۲ توصیف رفتار پویا

در این بخش به بیان معناشناسی پویای مدل در قالب معناشناسی عملیاتی خواهیم پرداخت. طبیعتاً لازم است معناشناسی ایستای مدل نیز توصیف شود که در اینجا برای رعایت اختصار از تشریح صوری آنها چشم‌پوشی می‌کنیم و صرفاً به طور غیررسمی به بیان دو فرض مهم بسنده می‌کنیم. اول این که تمامی نام‌ها شامل نام متغیرها، مؤلفه‌ها و سامانه‌ها یکتا هستند. دیگر این که فرض می‌شود در بیان قواعد، شرط‌ها و جملات انتساب از نظر نوع داده‌ها صحیح هستند.

پیش از تعریف صوری معناشناسی پویا نیاز است چند مفهوم را تعریف کنیم. فرضاً مجموعه Val تمامی مقداردهی‌های ممکن به متغیرهاست. $\text{effect} : Var \to Assign \to Val$ یک انتساب را در یک زمینه[1] مشخص انجام داده و ارزیابی متغیر مرتبط را تغییر می‌دهد. در این مقاله از نماد $\text{effect}_v(a)$ به جای $\text{effect}(v,a)$ استفاده خواهیم کرد. فرض می‌کنیم effect برای یک دنباله از انتساب‌ها نیز سربارگذاری[2] می‌شود: $\text{effect}_v(\langle a_1, a_2, \ldots, a_n \rangle) = \langle \text{effect}_v(a_1), \text{effect}_v(a_2), \ldots, \text{effect}_v(a_n) \rangle$. تابع $\text{trig}_v : Assign \to 2^{Event}$ مجموعه رخدادهایی را که در یک زمینه مشخص و با انجام یک انتساب مشخص فعال می‌شوند، برمی‌گرداند.

فرض می‌کنیم trig_v برای یک دنباله از انتساب‌ها سربارگذاری می‌شود: $\text{trig}_v(\langle a_1, a_2, \ldots, a_n \rangle) = \langle \text{trig}_v(a_1), \text{trig}_v(a_2), \ldots, \text{trig}_v(a_n) \rangle$. اگر در حالی که ارزشیابی یک رخداد غلط است، با یک انتساب در کنش‌های یک قاعده، ارزشیابی رخداد صحیح گردد، می‌گوییم که این رخداد فعال شده است.

با تابع $s : SName \to (Var \to Val) \times Actives^*$ حالت سراسری یک سامانه را نشان می‌دهیم که $Var \to Val$ یک نگاشت است که به متغیرهای مدل مقدار می‌دهد و $Actives$ یک دنباله از مجموعه‌های شامل رخدادهای فعال است. در یک حالت سراسری (v, θ)، نگاشت $v : Var \to Val$ مقداردهی متغیرها را مشخص می‌کند (v_0 ارزیابی اولیه متغیرها را نمایش می‌دهد) و θ_iهای موجود در دنباله θ که $1 \le i \le |\theta|$ است، نمونه‌هایی از نوع 2^{Clause} هستند.

گذارهای سامانه $(x, comp)$ را که مؤلفه‌های آن به صورت $comp = \langle comp_1, \ldots, comp_n \rangle$ است با قاعده‌های معناشناسی عملیاتی ساخت‌یافته در (۱) تا (۵) تعریف می‌کنیم. برای سهولت تعاریف، مجموعه تمامی قاعده‌های سامانه را با $R = \cup_{i=1}^{n} r_i$ نشان می‌دهیم. نمادهای $r.e$ و $r.a$ نیز به ترتیب نشان‌دهنده رخداد و کنش قاعده r هستند.

در دنباله رخدادهای فعال و در بین قواعد مربوط به یک مجموعه از رخدادهای فعال، قاعده بالاتر نوشته‌شده زودتر اجرا می‌شود. پس اگر داشته باشیم $r_p \in comp_w.r$ و $r_q \in comp_z.r$، اگر $w > z$ اولویت اجرا با r_q است، اگر $w < z$ اولویت اجرا با r_p است و اگر $w = z$ یعنی این دو قاعده در یک مؤلفه تعریف شده‌اند پس اگر $p > q$ اولویت اجرا با r_q و در غیر این صورت اولویت اجرا با r_p است.

معناشناسی سامانه گذار یک مدل پیشنهادی $\mathcal{U}$ را به صورت سه‌تایی $TS(\mathcal{U}) = (S \to s_0)$ تعریف می‌کنیم که در آن S مجموعه حالات سراسری، $\to$ رابطه تعریف‌شده در بالا و s_0 حالت اولیه سامانه را که به صورت (v_0, ϵ) است، تعریف می‌کنند.

۳- تولید خودکار آزمایه

رویکرد مورد استفاده در آزمون سامانه، تولید خودکار آزمایه مبتنی بر مدل رفتاری است. مدل رفتاری (که در بخش قبل درباره آن توضیح داده

1. Context
2. Overload

شد) می‌تواند به عنوان پیش‌گوی آزمون مورد استفاده قرار گیرد. با توجه به بزرگی نسبی سامانه‌های صنعتی مورد هدف این پژوهش، تولید آزمایه به روش‌هایی نظیر اجرای نمادین[1] قابل استفاده نیست و تولید آزمایه‌ها باید به صورت تصادفی انجام شود. با توجه به موفقیت نسبی روش‌های مبتنی بر جستجو در تولید خودکار آزمایه [۳۱]، در این پژوهش از الگوریتم ژنتیک برای تولید خودکار آزمایه استفاده کردیم. لازم به ذکر است که روش‌های فراابتکاری[2] متنوعی در تولید خودکار آزمایه استفاده شده‌اند که لزوماً یکی از آنها بر سایرین برتری مطلق ندارد [۳۲]. انتخاب الگوریتم ژنتیک به دلیل وجود تجربه‌های قبلی در تیم پژوهش و همچنین در دسترس بودن پیاده‌سازی‌های قابل اتکا برای این دسته الگوریتم‌ها صورت گرفته است. با توجه به دستیابی به نتایج مطلوب، الگوریتم‌های دیگر جستجو مورد مطالعه قرار نگرفتند؛ هرچند ممکن است بتوان با برخی از آنها به نتایجی با کارایی بالاتر نیز دست یافت.

نکته دیگر، انتخاب معیاری برای بسندگی آزمایه‌های تولیدشده است که در تابع برازش الگوریتم ژنتیک خود را نشان می‌دهد. بخش عمده‌ای از روش‌های جستجومحور به این منظور از معیارهای پوشش کد استفاده می‌کنند. با توجه به این که در سامانه‌های صنعتی، هر بار اجرای برنامه می‌تواند مستلزم فراهم‌کردن محیط مناسب برای اجرای برنامه باشد (نظیر محیط سخت‌افزاری لازم برای یک نرم‌افزار نهفته)، اجرای برنامه به ازای هر کروموزم تولیدشده بسیار پرهزینه خواهد بود. ما برای رفع این مشکل با اتکا به داشتن مدل رفتاری سامانه، بسندگی آزمایه‌ها را بر مبنای پوشش مدل رفتاری تعریف می‌کنیم. به این ترتیب، علاوه بر پوشش نیازمندی‌های سامانه (که در قالب مدل رفتاری متجلی شده است) می‌توانیم به کارایی نسبتاً بالاتری دست پیدا کنیم و استفاده از این روش را در عمل ممکن سازیم. لازم به ذکر است که استفاده از پوشش مدل در سامانه‌های صنعتی به طور موفقی در دامنه سامانه‌های مالی نیز مورد استفاده قرار گرفته است [۹]. در ادامه با معرفی دو معیار پوشش قاعده و شرط فعال قاعده نشان خواهیم داد که چگونه این معیارها در تعریف تابع برازش الگوریتم جستجو مورد استفاده قرار می‌گیرند.

۳-۱ نیازمندی‌های آزمون

همان گونه که اشاره شد، معیارهای انتخاب آزمایه پیشنهادی بر اساس پوشش روی قواعد تعریف شده‌اند. در انتخاب این معیارها به کاربردی‌بودن آنها در محیط صنعتی توجه شده است. مثلاً اگر معیاری انتخاب شود که مجموعه آزمون بیش از اندازه بزرگی تولید کند، فرایند آزمون را دچار اختلال خواهد کرد؛ زیرا اجرای آن بیش از اندازه زمان‌بر خواهد بود.

فرض کنیم R مجموعه قواعد توصیف، C مجموعه شروط یک قاعده و TR مجموعه نیازمندی‌های آزمون معیار پوشش را نشان بدهند. برای پوشش قاعده به ازای اجرای هر قاعده $r \in R$ یک نیازمندی آزمون در مجموعه نیازمندی‌های آزمون TR خواهیم داشت که به معنای اجراشدن آن قاعده در جریان آزمون است. از آنجا که ممکن است اجرای یک قاعده تحت شرایط متفاوتی انجام شود، معیار پوشش قاعده هرچند در عمل به سادگی قابل دسترسی است، اما لزوماً مسیرهای اجرایی مختلف سیستم را تحت پوشش قرار نمی‌دهد. به همین دلیل معیارهای مبتنی بر شرط‌های قاعده‌ها را تعریف خواهیم کرد که مبتنی بر معیارهای پوشش منطقی هستند که پیش از این در حوزه آزمون نرم‌افزار مورد مطالعه قرار گرفته‌اند [۳۳]. به این هدف می‌توان مجموعه شرط‌های یک قاعده را یک محمول[3] و هر شرط را یک بند[4] دانست. در مدل پیشنهادی، ترکیب عطفی مجموعه شرط‌های C در نظر گرفته می‌شود؛ بنابراین تمامی شرط‌های یک قاعده می‌توانند شرط اصلی باشند. همین طور یک شرط $c \in C$، فقط در صورت صحیح‌بودن ارزیابی سایر شرط‌ها (بندها)، شرط اصلی محسوب می‌شود؛ زیرا در این صورت c صحیح یا غلط بودن ارزیابی نهایی C را تعیین می‌کند. برای پوشش شرط فعال قاعده، به ازای فعال‌شدن رخداد هر قاعده $r \in R$ و هر شرط اصلی c_i در C شرط‌های فرعی c_j $(i \neq j)$ را طوری مقداردهی می‌کنیم که ارزیابی صحیح داشته باشند تا c_i ارزیابی نهایی C را تعیین کند. به ازای هر c_i دو نیازمندی آزمون در TR داریم: مقدار ارزیابی c_i صحیح باشد و مقدار ارزیابی c_i نادرست باشد. به عنوان مثال برای قاعده produce در مدل ارائه‌شده در مدل دستگاه فروش خودکار، سه نیازمندی آزمون پوشش قاعده فعال خواهیم داشت:

۱) رخداد command = produce فعال شود و paymentResult = true و orderState = selected باشد.

۲) رخداد command = produce فعال شود و paymentResult != true و orderState = selected باشد.

۳) رخداد command = produce فعال شود و paymentResult = true و orderState != selected باشد.

نیازمندی آزمون ۱، عملاً همان نیازمندی آزمون معیار پوشش قاعده برای این قاعده است. به طور کلی در نیازمندی‌های آزمون پوشش شرط فعال یک قاعده، نیازمندی آزمون پوشش قاعده آن نیز قرار دارد. بنابراین معیار پوشش شرط فعال قاعده، شامل معیار پوشش قاعده می‌شود.

۳-۲ تولید جستجومحور آزمایه‌ها

در این مرحله با تبدیل صورت مسئله به یک مسئله بهینه‌سازی و با استفاده از الگوریتم ژنتیک، اقدام به یافتن کوتاه‌ترین مجموعه آزمون با بیشترین پوشش می‌کنیم. در این مسئله، یک آزمایه از یک جفت ورودی و خروجی تشکیل می‌شود و مجموعه آزمون، یک دنباله از جفت‌های ورودی و خروجی است. منظور از ورودی در اینجا، تغییر مقدار یک متغیر ورودی سامانه است. یک ورودی ممکن است موجب فعال‌شدن هیچ رخدادی نشود؛ اما منظور از خروجی، تغییر مقدار صفر یا بیشتر متغیر خروجی سامانه است؛ زیرا با دریافت یک ورودی بر اساس قاعده‌های سامانه، صفر یا بیشتر متغیر خروجی تغییر مقدار خواهند داشت.

برای تبدیل دنباله ورودی‌های آزمون به یک کروموزوم، ورودی‌های سیستم کدگذاری می‌شوند و از این کدها به عنوان عناصر کروموزوم استفاده می‌شود. جدول ۱ کدگذاری ورودی‌های دستگاه فروش خودکار را نشان می‌دهد.

از آنجا که الگوریتم مورد استفاده طول کروموزوم‌ها را ثابت فرض می‌کند، یک طول ثابت و مطمئن (با استفاده از دانش دامنه توصیف) تعیین می‌کنیم که آن را ثابت بیشینه طول ورودی‌های آزمون (یا بیشینه اندازه مجموعه آزمون) می‌نامیم. اما برای این که بتوانیم در عمل طول مجموعه آزمون را کمتر کنیم، یک کد «پوچ» به مجموعه کدهای ورودی‌ها اضافه می‌کنیم که عملاً معادل ورودی‌ندادن به سیستم است. هنگام آزمون نیز این ورودی‌ها را نادیده می‌گیریم. به این ترتیب می‌توانیم

1. Symbolic Execution
2. Metaheuristic
3. Predicate
4. Clause

جدول ۱: کدگذاری ورودی‌های دستگاه فروش خودکار.

کد	ورودی
۰	command = coffee
۱	command = tea
۲	command = sugar
۳	command = produce
۴	command = take
۵	command = clear
۶	money = true
۷	none

علاوه بر افزایش پوشش آزمایه‌ها به کوچک‌ترکردن اندازه آنها نیز توجه کرد. به این منظور تابع برازشی تعریف می‌کنیم که در آن مقدار بیشتر به معنای بهتربودن آن است: $f(\overline{x}) = (L+1)\times c(\overline{x}) - s(\overline{x})$. در این تابع $c(\overline{x})$ پوشش به‌دست‌آمده با دنباله ورودی‌های آزمون $\overline{x}$، L ثابت بیشینه طول ورودی‌های آزمون که در ابتدا به عنوان ورودی الگوریتم تعیین می‌شود و $s(\overline{x})$ تعداد ورودی‌های $\overline{x}$ (بدون در نظر گرفتن ورودی‌های پوچ) را نشان می‌دهند. از آنجا که افزایش پوشش بر کاهش تعداد آزمایه‌ها اولویت دارد می‌توان در تابع برازش f با استفاده از ضریب، این اولویت را نشان داد و نیازی به استفاده از بهینه‌سازی پرتو[1] نیست.

جدول ۲ یک پاسخ نمونه را نشان می‌دهد که در آن کدهای ورودی‌ها به ترتیب اجرا در یک آرایه قرار گرفته‌اند. بیشینه اندازه مجموعه آزمون در این مسئله ۱۵ در نظر گرفته شده است. این پاسخ کاندیدای ارائه‌شده در ابتدا ورودی با کد ۶ یعنی money = true، سپس ورودی با کد ۰ یعنی command = coffee و به همین ترتیب ورودی‌های متناظر با شناسه‌های بعدی را نشان می‌دهد. در نهایت با حذف ورودی‌های پوچ (کد ۷)، اندازه مجموعه آزمون حاصل ۸ خواهد بود. به عبارت دیگر این توالی ورودی ۸ ورودی غیرپوچ دارد. جدول ۳ مجموعه آزمون متناظر با این توالی ورودی برای معیار پوشش قاعده را نمایش می‌دهد که در آن خروجی متناظر با هر ورودی و قاعده اجراشده توسط آن نوشته شده‌اند.

٤- مطالعه موردی و ارزیابی

برای ارزیابی روش پیشنهادی، آن را در یک مطالعه موردی در شرکت کروز که بزرگ‌ترین تولیدکننده قطعات برقی خودرو در ایران است به کار بردیم. در فرایند توسعه نرم‌افزار در این شرکت که مبتنی بر مدل وی [۳۴] است، نیازمندی‌های سطح بالا طی مراحلی به سندی به نام «توصیف نیازمندی‌های طراحی (DRS)[2]» تبدیل می‌شوند که ساختاری قاعده‌محور دارند و در قالب جدول‌های صفحه‌گسترده توصیف می‌شوند. فرایند طراحی آزمایه‌ها در این شرکت به صورت دستی از روی این توصیف‌ها صورت می‌گیرد. در این مطالعه موردی که روی واحد پایشگر[3] از واحد کنترل موتور (ECU)[4] انجام شد، تولید آزمایه‌ها به صورت خودکار از روی توصیف نیازمندی‌های طراحی صورت گرفت.

٤-١ مدل‌سازی رفتار سامانه

در مرحله اول این مطالعه موردی لازم بود رفتار سیستم که در اسناد توصیف نیازمندی‌های طراحی مشخص گردیده بود، به مدل صوری پیشنهادی تبدیل شود. طی این تبدیل موارد متعددی از نقص‌ها یا ابهام‌های موجود در این اسناد کشف شد که نشان‌دهنده اهمیت‌داشتن پشتوانه صوری در توصیف رفتار سیستم است. این موارد طی تعامل با طراحان سیستم رفع شدند و نهایتاً رفتار صحیح سیستم در قالب مدل پیشنهادی توصیف شد. به علت حفظ محرمانگی اطلاعات، امکان ارائه جزئیات توصیف‌های سیستم مورد بحث وجود ندارد. با این حال، یک مثال و توضیح کوتاه از نمایش جدولی توصیف نیازمندی‌های طراحی ارائه شده است. توصیف نیازمندی‌های طراحی از جداولی تشکیل شده که هر سطر آنها یک قاعده مبتنی بر حالت و رخداد را بیان می‌کند. با افزایش حجم توصیف، نمایش جدولی به خواناترشدن آن کمک می‌کند. در شرکت کروز این جداول در ابزار Rational DOORS نگهداری می‌شوند که یکی از پراستفاده‌ترین ابزارهای نگهداری و ویرایش توصیف نیازمندی‌ها در صنعت خودرو است.

هر زیرسامانه در واحد پایشگر دارای چندین جدول است که توصیف نیازمندی‌های طراحی آن را در ابعاد مختلف تعریف می‌کنند. جدول ۴ دو قاعده از یکی از زیرسامانه‌های واحد پایشگر را نشان می‌دهد که به صورت جدولی نوشته شده‌اند و در ادامه به صورت خلاصه به توضیح خصوصیات این مثال می‌پردازیم.

ستون SRS_ID شناسه یا نام قواعد را مشخص می‌کند و ستون‌های بعدی هر کدام به یک متغیر و نقش آن در قاعده‌ها اختصاص دارند که به دو دسته کلی تقسیم می‌شوند: ستون‌های شرط یا رخداد و ستون‌های کنش. در صورتی که از یک متغیر در هر دو بخش استفاده شود، پیش از نام آن در بخش اول از _Pre استفاده می‌گردد؛ مانند TimerNdisEn. مقادیری که متعلق به متغیرهای شرط هستند با پیشوند _S و مقادیری که متعلق به متغیرهای رخداد و کنش هستند با پیشوند _T نمایش داده می‌شوند. به علت نمایش چندین قاعده در یک جدول، یک متغیر ممکن است تنها در برخی از قواعد شرکت داشته باشد؛ اما با این حال نیاز است تا یک ستون به آن اختصاص یابد. به متغیری که در قاعده تأثیری نداشته باشد، اگر در برخی قواعد دیگر جدول نقش رخداد و یا شرط را ایفا کند، مقدار N/A و اگر در برخی قواعد دیگر جدول نقش کنش داشته باشد، مقدار NO_CHANGE نسبت داده می‌شود. لازم به ذکر است که جدول‌های توصیف نیازمندی‌های طراحی معمولاً اندازه بزرگی دارند که در اینجا به دلیل کمبود فضا از نمایش آنها خودداری کردیم. تبدیل این دو قاعده به مدل صوری پیشنهادی در زیر آمده است.

```
PSH_0003 {
   WHEN VAR_OSM_OpState == NORMAL_ST
   WHILE VAR_RSTH_CntRstPrimary >= 2
   THEN TimerNdisEn = START
}

PSH_0006 {
   WHEN TimerNdisEn == TIMEOUT
   WHILE VAR_OSM_OpState == NORMAL_ST
   THEN LV_PSH_OsmExpectedLsdNabe = ENABLE,
      LV_PSH_LsdNabe = ENABLE,
      LV_PSH_EtcEn = ENABLE
}
```

سامانه مطالعه‌شده (واحد پایشگر) شامل ۱۳ زیرسامانه است که مجموعاً شامل ۴۸۵ قاعده، ۳۶۸ ورودی و مجموعاً ۲۰۷ متغیر (ورودی، خروجی و داخلی) است.

1. Pareto Optimization
2. Design Requirements Specifications
3. Monitoring Unit
4. Engine Control Unit

جدول ۲: یک پاسخ کاندیدا (توالی آزمون) در الگوریتم ژنتیک.

کد ورودی	۶	۰	۲	۷	۷	۳	۷	۴	۱	۷	۷	۵	۰	۷

جدول ۳: مجموعه آزمون متناظر با توالی ورودی آزمون ارائه‌شده در جدول ۲.

کد ورودی	ورودی	قواعد اجراشده	خروجی
۶	money = true	pay	–
۰	command = coffee	selectCoffee	–
۲	command = sugar	addSugar	–
۳	command = produce	produce	drink = coffeeWithSugar
۴	command = take	take و ready	–
۱	command = tea	selectTea	–
۵	command = clear	clear	–
۰	command = coffee	selectCoffee	–

جدول ۴: نمایش جدولی دو قاعده در توصیف واحد پایشگر (به شکل خلاصه‌شده).

SRS_ID	VAR_OSM_ OpState	VAR_RSTH_ CntRstPrimary	Pre_ TimerNdisEn	TimerNdisEn	LV_PSH_ OsmExpectedLsdNabe	LV_PSH_ LsdNabe	LV_PSH_ EtcEn
PSH_۰۰۳	T_NORMAL_ST	S_MTE_۲	N/A	T_START	NO_CAHNGE	NO_CAHNGE	NO_CAHNGE
PSH_۰۰۶	S_NORMAL_ST	N/A	T_TIMEOUT	NO_CAHNGE	T_ENABLE	T_ENABLE	T_ENABLE

٤-۲ تولید آزمایه

در این پژوهش از الگوریتم ژنتیک نخبه‌گرا با کمک کتابخانه geneticalgorithm[1] در زبان پایتون استفاده شد. جمعیت مورد استفاده در این الگوریتم ۱۰۰ کروموزوم است و احتمال جهش، نسبت نخبگان و نسبت والدین هر سه ۱۰ درصد تعیین شده‌اند. احتمال تقاطع نیز ۵۰ درصد در نظر گرفته شده است. عملگر تقاطع مورد استفاده نیز تقاطع دونقطه است. از آنجا که مقدار برازش بهینه از پیش مشخص نیست و به علت گستردگی فضای حالت، از تعداد تکرار مشخص برای الگوریتم به عنوان شرط پایان آن استفاده می‌کنیم. در انتخاب این مشخصه باید توجه داشت که یکی از مهم‌ترین عامل‌های تعیین‌کننده زمان اجرای الگوریتم ژنتیک، شرط پایان آن است و تعیین تعداد تکرارهای بالا، افزایش زمان اجرای الگوریتم را در پی خواهد داشت. از سوی دیگر انتخاب تعداد تکرارهای بیش از حد کم موجب خواهد شد که الگوریتم فرصت یافتن جواب بهینه را پیدا نکند. با توجه به مشاهدات برای اجراهای مربوط به پوشش قاعده از ۲۰۰ تکرار و برای اجراهای مربوط به پوشش شرط فعال قاعده از ۳۰۰ تکرار استفاده شد.

تعیین پارامترهای الگوریتم به صورت تجربی انجام شده است. به عنوان مثال در اجراهای اولیه با جمعیتی با اندازه ۱۰۰۰، اجرای الگوریتم را شروع کردیم و با توجه به نتایج به‌دست‌آمده به‌تدریج این تعداد را کم کردیم. با کاهش جمعیت، زمان اجرای مورد نیاز الگوریتم کاهش پیدا می‌کند؛ اما تا حدود جمعیت ۱۰۰، کاهش چشم‌گیری در پوشش به‌دست‌آمده ایجاد نشد. به طور مشابه در خصوص زمان اجرای الگوریتم (که توسط تعداد تکرار تعیین می‌شود)، تجربه نشان داد افزایش زمان اجرا بیش از حدود ۴/۵ ساعت (برای شرط فعال قاعده) منجر به دستیابی به پوشش بیشتر نخواهد شد. لازم به ذکر است ما در اجرای الگوریتم سقف زمانی یک روز کاری (حدود ۸ ساعت) را داشتیم و امکان ادامه اجرای الگوریتم بیش از آن وجود نداشت. نکته مهم این است که این مقادیر به طور مطلق و مستقل از سامانه تحت آزمون از پیش قابل تعیین نیستند و به‌کارگیری روش پیشنهادی در کاربردهای دیگر مستلزم انجام چنین تنظیماتی به صورت تجربی خواهد بود.

٤-۳ نتایج

برای اجرای فرایند طراحی آزمون روی توصیف واحد پایشگر، روش پیشنهادی روی ۱۳ زیرسیستم واحد پایشگر اجرا شد. میانگین به‌دست‌آمده برای ۳ بار اجرای الگوریتم ژنتیک با معیار پوشش قاعده در جدول ۵ مشخص گردیده که طبق آن در ۱۱ زیرسامانه پوشش کامل و در ۲ زیرسامانه به ترتیب پوشش‌های ۴/۹۴ و ۴/۹۷ درصد به دست آمد. در مجموع این زیرسامانه‌ها ۸/۹۸ درصد قواعد در این مجموعه آزمون پوشیده شدند و از بین مجموع ۴۸۹ قاعده، ۶ قاعده در این مجموعه آزمون اجرا نشدند. این تعداد قاعده با ۲۲۸۷ آزمایه پوشش می‌شوند که تعدادی مناسب برای اجرای عملی آنها بر روی سامانه تحت آزمون است.

برخی نیازمندی‌های آزمون پیچیدگی بسیاری دارند و احتمال پوشش آنها توسط الگوریتم‌های تولید آزمون بسیار پایین است. به عنوان مثال در این مطالعه موردی، زیرسامانه سه، دارای چهار قاعده است که رخدادهای آنها تنها در صورتی فعال می‌شوند که یک ورودی خاص، شش بار متوالی وارد شود (به عبارت دیگر برای اجرای هر یک از این قواعد باید در یک شرایط خاص یک ورودی خاص شش بار متوالی وارد شود). الگوریتم ژنتیک، موفق به پوشش این چهار قاعده در اجرای روش با پوشش قاعده و دوازده نیازمندی آزمون مرتبط با آنها در اجرای روش با پوشش شرط فعال قاعده نشد. تجربه نشان داد روش پیشنهادی با افزایش جمعیت یا افزایش مدت اجرای الگوریتم در مدت معقول قادر به پوشش این موارد خاص نیست. برای رفع این موضوع باید تابع برازش الگوریتم طوری تغییر یابد که جمعیت را به سمت بخش‌های پوشش‌نیافته هدایت کند. تعریف چنین توابعی در دستور کار ما برای پژوهش‌های آینده است.

نکته دیگری که باید به آن اشاره کرد، زمان اجرای مربوط به این دو ارزیابی است. ارزیابی اول که مربوط به اجرای فرایند پیشنهادی با معیار پوشش قاعده بود، در یک ساعت و پانزده دقیقه انجام پذیرفت و ارزیابی

1. https://pypi.org/project/geneticalgorithm

جدول ۵: نتایج تولید خودکار آزمایه برای زیرسامانه‌های واحد پایشگر.

شماره زیرسامانه	بیشینه اندازه مجموعه آزمون	اندازه مجموعه آزمون پاسخ	تعداد قواعد	تعداد قواعد پوشش‌داده‌شده	درصد پوشش
۱	۴۰	۳۰	۲۰	۲۰	۱۰۰
۲	۴۰	۲۴	۱۷	۱۷	۱۰۰
۳	۳۰۰	۲۸۳	۶۷	۶۳	۹۴٫۴
۴	۱۵۰	۱۳۸	۶۳	۶۳	۱۰۰
۵	۱۰۰	۹۰	۵۲	۵۲	۱۰۰
۶	۱۰۰۰	۹۸۰	۷۹	۷۷	۹۷٫۴
۷	۱۰۰	۹۰	۳۴	۳۴	۱۰۰
۸	۴۰	۲۶	۱۷	۱۷	۱۰۰
۹	۲۰	۱۷	۱۶	۱۶	۱۰۰
۱۰	۳۰	۱۷	۱۰	۱۰	۱۰۰
۱۱	۵۰	۳۳	۱۴	۱۴	۱۰۰
۱۲	۱۰۰	۷۷	۴۱	۴۱	۱۰۰
۱۳	۵۰۰	۴۸۲	۵۹	۵۹	۱۰۰
	مجموع	۲۲۸۷	۴۸۹	۴۸۳	۹۸٫۷

دوم که مربوط به اجرای این فرایند با معیار پوشش شرط فعال قاعده بود در حدود ۴٫۵ ساعت به طول انجامید. در این ارزیابی از زبان برنامه‌نویسی پایتون، سیستم‌عامل ویندوز ۱۰، پردازنده Core i5 و هشت گیگابایت حافظه استفاده شده و این در حالی است که فرایند طراحی آزمون دستی واحد پایشگر هفته‌ها به طول می‌انجامد. بنابراین پیاده‌سازی روش پیشنهادی و ادغام آن در فرایند آزمون شرکت، هزینه تولید آزمایه‌ها را به شکل قابل توجهی کاهش می‌دهد.

۵- نتیجه‌گیری

در این مقاله یک روش نوین آزمون مبتنی بر مدل جعبه سیاه ارائه شد که برای تولید خودکار آزمایه برای نرم‌افزارهای نهفته در مقیاس صنعتی قابل استفاده است. به این منظور یک زبان مدل‌سازی رفتار قاعده‌محور ارائه گردید که می‌تواند مدل رفتاری سامانه را به شکلی مؤلفه‌محور به نحوی موجز بیان کند. با به‌کارگیری یک رویکرد جستجومحور مبتنی بر الگوریتم ژنتیک می‌توان بر مبنای این مدل آزمایه تولید کرد. الگوریتم جستجو بر مبنای بهینه‌سازی معیارهای پوشش قاعده‌محور که در این مقاله تعریف شدند، عمل می‌کند. اثربخشی روش ارائه‌شده با به‌کارگیری موفق آن روی یک نرم‌افزار نهفته صنعتی واقعی مورد بررسی قرار گرفت. با توجه به این که در دامنه نرم‌افزارهای نهفته (به‌خصوص در کاربردهای حساس مثل کنترل اجزای خودرو) مدل‌سازی رفتاری نرم‌افزار جزئی از فرایندهای متداول تیم‌های توسعه‌دهنده است و مدل‌سازی سامانه سربار برای توسعه نرم‌افزار محسوب نمی‌شود، روش پیشنهادی به طور مؤثری در عمل قابل استفاده است. شایان ذکر است این که زبان مدل‌سازی ارائه‌شده از نحوی ساده و قابل استفاده توسط مهندسین برخوردار است و بدون نیاز به آموزش خاصی در حوزه روش‌های صوری می‌توان از روش پیشنهادی در پروژه‌های صنعتی استفاده نمود.

به عنوان قدم‌هایی برای ادامه این پژوهش می‌توان روی تعریف توابع برازش کار کرد؛ به طوری که به نحو مؤثرتری جستجو را به سمت بخش‌های پوشش‌نیافته هدایت کنند. به علاوه می‌توان به عنوان الگوریتم جستجو از روش‌های فراابتکاری دیگری به‌جز الگوریتم ژنتیک استفاده کرد و میزان اثربخشی و زمان اجرای لازم را مطالعه نمود. نهایتاً استفاده از پیاده‌سازی‌های کاراتر الگوریتم ژنتیک برای کم‌کردن زمان اجرای آزمون‌ها می‌تواند مورد توجه قرار گیرد.

مراجع

[1] J. Zander, Model-Based Testing of Real-Time Embedded Systems in the Automotive Domain, Ph.D Thesis, Technical University of Berlin, 2009.

[2] R. N. Charette, "This car runs on code," *IEEE Spectrum*, vol. 46, no. 3, p. 3, Feb. 2009.

[3] J. Babic, M. Siniša, and I. Petrovic, "Introducing model-based techniques into development of real-time embedded applications," *Automatika*, vol. 52, no. 4, pp. 329-338, Oct. 2011.

[4] A. Kramer and B. Legeard, *2019 Model-Based Testing User Survey: Results*, Technical Report, Comité Français des Tests Logiciels, Jan. 2020.

[5] W. Li, F. Le Gall, and N. Spaseski, "A survey on model-based testing tools for test case generation," in *Proc. 4th Int. Conf. on Tools and Methods of Program Analysis*, pp. 77-89, Moscow, Russia, 3-4 Mar. 2017.

[6] A. Shaout and S. Pattela, "Model based approach for automotive embedded systems," in *Proc. 22nd Int. Arab Conf. on Information Technology*, 7 pp., Muscat, Oman, 21-23 Dec. 2021.

[7] M. N. Zafar, W. Afzal, and E. Enoiu, "Evaluating system-level test generation for industrial software: a comparison between manual, combinatorial and model-based testing," in *Proc. of the 3rd ACM/IEEE Int. Conf. on Automation of Software Test*, pp. 148-159, Pittsburgh, PA, USA, 21-22 May 2022.

[8] V. Garousi, A. B. Keleş, Y. Balaman, Z. Özdemir Güler, and A. Arcuri, "Model-based testing in practice: an experience report from the web applications domain," *J. of Systems and Software*, vol. 180, Article ID: 111032, Oct. 2021.

[9] A. Zakeriyan, R. Khosravi, H. Safari, E. Khamespanah, and S. M. Shamsabadi, "Automated testing of an industrial stock market trading platform based on functional specification," *Science of Computer Programming*, vol. 225, Article ID: 102908, Jan. 2023.

[10] H. R. Asaadi, R. Khosravi, M. Mousavi, and N. Noroozi, "Towards model-based testing of electronic funds transfer systems," in *Proc. Int. Conf. on Fundamentals of Software Engineering,* pp. 253-267, Tehran, Iran, 20-22 Apr. 2011.

[11] J. Tretmans, "Test generation with inputs, outputs and repetitive quiescence," *Software-Concepts and Tools*, vol. 17, no. 3, pp. 103-120, 1996.

[12] M. van der Bijl, A. Resnik, and J. Tretmans, "Compositional testing with ioco," in *Proc. Third Int. Workshop on Formal Approaches to Testing of Software*, pp. 86-100, Montreal, Canada, 6-6 Oct. 2003.

[13] P. Daca, T. Henzinger, W. Krenn, and D. Nickovic, "Compositional specifications for ioco testing," in *Proc. IEEE 7th Int. Conf. on Software Testing, Verification and Validation*pp. 373-382, Cleveland, OH, USA, 31 Mar.-4 Apr. 2014.

[14] M. L. Mohd-Shafie, *et al.*, "An EFSM-based test data generation approach in model-based testing," *Computers, Materials & Continua*, vol. 71, no. 3, pp. 4337-4354, Jan. 2022.

[15] W. Huang, N. Krafczyk, and J. Peleska, "Exhaustive property-oriented model-based testing with symbolic finite state machines,"

Science of Computer Programming, vol. 231, Article ID: 103005, Aug. 2023.
[16] N. Van Kelecom, S. Silverans, and M. Dutre, "A traceable development and testing methodology for embedded software," in *Proc. 12th Graz Symp. on Virtual Vehicle*, 8 pp., Graz, Austria, 7-8 May 2019.
[17] M. L. Mohd-Shafie, W. M. N. Wan-Kadir, H. Lichter, M. Khatibsyarbini, and M. Adham-Isa, "Model-based test case generation and prioritization: a systematic literature review," *Software and Systems Modeling*, vol. 21, no. 2, pp. 751-753, Apr. 2022.
[18] R. Ferdous, C. Hung, F. Kifetew, D. Prandi, and A. Susi, "EvoMBT: evolutionary model-based testing," *Science of Computer Programming*, vol. 227, Article ID: 102942, Mar. 2023.
[19] U. C. Türker, R. M. Hierons, K. El-Fakih, M. R. Mousavi, and I. Y. Tyukin, "Accelerating finite state machine-based testing using reinforcement learning," *IEEE Trans. on Software Engineering*, vol. 50, no. 3, pp. 574-597, Mar. 2024.
[20] -, intel/fMBT: Free Model Based tool. URL: https://github.com/intel/fMBT, accessed: Dec. 2022.
[21] M. N. Zafar, *et al.*, "Model-based testing in practice: an industrial case study using graphwalker," in *Proc. 14th Innovations in Software Engineering Conf.*, 11 pp., Bhubaneswar, India, 25-27 Feb. 2021.
[22] V. Aravantinos, S. Voss, S. Mavin, F. Hölzl, and B. Schätz, "AutoFOCUS 3: Tooling Concepts for Seamless, Model-based Development of Embedded Systems", in *Joint Proc. of the 8th Int. Workshop on Model-based Architecting of Cyber-physical and Embedded Systems and 1st Int. Workshop on UML Consistency Rules*, pp. 19-26, Ottawa, Canada, 28-28 Sept. 2015.
[23] J. Tretmans and P. van de Laar, "Model-based testing with torXakis," in *Proc. 30th Central European Conf. on Information and Intelligent Systems*, pp. 247-258, Varaždin, Croatia, 2-4 Oct. 2019.
[24] P. Kaur and G. Gupta, "Automated model-based test path generation from UML diagrams via graph coverage tech-niques," *International J. of Computer Science and Mobile Computing*, vol. 2, no. 7, pp. 302-311, Jul. 2013.
[25] A. Huima, "Implementing conformiq qtronic," in *Joint Proc. of 19th IFIP TC 6/WG 6.1 Int. Conf., TestCom 2007, and 7th Int. Workshop Testing of Software and Communicating Systems*, 12 pp., Tallin, Estonia, 26-29 Jun. 2007.
[26] G. J. Holzmann, *The Spin Model Checker: Primer and Reference Manual*, Addison-Wesley, 2004.
[27] *Rebeca Modeling Language*, https://rebeca-lang.org, accessed Feb. 2024.
[28] *Event-B*, https://www.event-b.org, accessed Feb. 2024.
[29] M. Atif and J. F. Groote, *Understanding Behaviour of Distributed Systems Using mCRL2*, *Springer*, 2023.
[30] A. Mavin, P. Wilkinson, A. Harwood, and M. Novak, "Easy approach to requirements syntax (EARS)," in *Proc. 17th IEEE Int. Requirements Engineering Conf.*, pp. 317-322, Atlanta, GA, USA, 31 Aug.-4 Sep. 2009.
[31] P. McMinn, "Search-based software testing: past, present and future," in *Proc. IEEE 4th Int. Conf. on Software Testing, Verification and Validation Workshops*, pp. 153-163, Berlin, Germany, 21-25 Mar. 2011.
[32] M. Khari and P. Kumar, "An extensive evaluation of search-based software testing: a review," *Soft Computing*, vol. 23, pp. 1933-1946, Mar. 2019.
[33] P. Ammann and J. Offutt, *Introduction to Software Testing*, 2nd Ed., Cambridge University Press, 2016.
[34] K. Forsberg and H. Mooz, "The relationship of systems engineering to the project cycle," *Engineering Management J.*, vol. 4, no. 3, pp. 36-43, 1992.

علی حبیبی مدرک کارشناسی مهندسی کامپیوتر خود را از دانشگاه خوارزمی در سال ۱۳۹۶ و مدرک کارشناسی ارشد مهندسی کامپیوتر خود را از دانشگاه تهران در سال ۱۴۰۰ دریافت کرد. در طول تحصیل در مقطع کارشناسی ارشد، او به عنوان عضوی از آزمایشگاه روش‌های صوری دانشکده مهندسی کامپیوتر دانشگاه تهران در طرح پژوهشی کاربردی این آزمایشگاه با شرکت کروز با موضوع آزمون مبتنی بر مدل نرم‌افزارهای مورد استفاده در خودرو مشارکت داشت. زمینه‌های پژوهشی مورد علاقه نام‌برده آزمون نرم‌افزار و مدل‌سازی و درستی‌سنجی صوری سیستم‌ها است.

رامتین خسروی دانش‌آموخته رشته مهندسی کامپیوتر دانشگاه صنعتی شریف در مقاطع کارشناسی در سال ۱۳۷۴، کارشناسی ارشد در سال ۱۳۷۶ و دکتری در سال ۱۳۸۴ است. او از سال ۱۳۸۶ عضو هیأت علمی گروه نرم‌افزار دانشکده مهندسی برق و کامپیوتر دانشکدگان فنی دانشگاه تهران است و تأسیس آزمایشگاه‌های پژوهشی معماری نرم‌افزار و تصدیق کیفیت نرم‌افزار این دانشکده را برعهده دارد. زمینه‌های پژوهشی مورد علاقه نام‌برده حوزه‌های مرتبط با مهندسی نرم‌افزار و به طور خاص، آزمون نرم‌افزار، مدل‌سازی و درستی‌سنجی صوری سیستم‌های توزیع‌شده و معماری نرم‌افزار است.