تعیین سطح توانایی روش‌های رسمی در توصیف و تحلیل نرم‌افزار

سیدمرتضی بابامیر و ویدا احمدی ثابت

چکیده: توسعه‌دهندگان نرم‌افزار به دلایل مختلف با مشکل انتخاب روش رسمی متناسب با نرم‌افزار تحت توسعه روبه‌رو هستند. هدف ما در این مقاله تعیین سطح توانایی روش‌های رسمی برای توصیف و تحلیل نرم‌افزارهای مختلف در چهار قدم است: در قدم اول معیارهایی که روش‌های رسمی با آنها سنجیده می‌شوند، معرفی می‌شوند. در قدم‌های دوم و سوم انواع نرم‌افزارها و روش‌های رسمی بر اساس رویکردشان در حل مسئله طبقه‌بندی می‌شوند و در قدم چهارم بر اساس معیارهای تعیین‌شده در قدم اول، برازندگی و تناسب چند نمونه از روش‌های رسمی برای توصیف و تحلیل هر طبقه از نرم‌افزار تعیین می‌شود.

کلید واژه: توصیف و وارسی نرم‌افزار، روش‌های رسمی، طبقه‌بندی نرم‌افزار، مبتنی بر حالت، مبتنی بر رخداد.

۱- مقدمه

سیستم‌های سخت‌افزاری و نرم‌افزاری از نظر نوع عملکرد و مقیاس در حال رشد و توسعه هستند؛ به دلیل این توسعه که باعث افزایش پیچیدگی این سیستم‌ها می‌شود، احتمال بروز خطاهای پیچیده نیز گسترش می‌یابد. علاوه بر این، به‌وجود آمدن چنین خطاهایی منجر به از دست رفتن منابع مالی، زمان، یا حتی زندگی انسان‌ها می‌شود.

هدف اصلی مهندسین نرم‌افزار، کمک به توسعه‌دهندگان نرم‌افزار در جهت ساخت نرم‌افزارهایی است که علی رغم پیچیدگی، به‌صورت قابل اطمینان کار کنند. یک راه حل برای رسیدن به این هدف استفاده از روش‌های رسمی[1] می‌باشد. روش‌های رسمی، روش‌هایی هستند که بر پایه ریاضی بوده و می‌توانند شامل زبان، تکنیک یا ابزارهایی باشند که برای توصیف و وارسی نرم‌افزار مورد استفاده قرار گیرند. به‌طور غیر رسمی، فرایند تشریح نرم‌افزار و ویژگی‌های مطلوب آن به‌وسیله زبان‌های رسمی، توصیف و فرایند بررسی و تحلیل یک نرم‌افزار جهت داشتن ویژگی‌های مطلوب آن، وارسی نامیده می‌شود [۱].

زبان‌های توصیف نرم‌افزار به‌صورت کلی به سه دسته اصلی تقسیم می‌گردند [۲]: زبان‌های رسمی، زبان‌هایی هستند که از نظر نحوی و معنایی عاری از ابهام هستند (مانند زبان Z)، زبان‌های نیمه‌رسمی[2] (مانند UML[3]) که از نظر نحوی عاری از ابهام ولی از لحاظ معنایی دارای ابهام می‌باشند و زبان‌های غیر رسمی (مانند زبان محاوره) که هم از لحاظ معنایی و هم از لحاظ نحوی دارای ابهام هستند.

توصیف و تحلیل نرم‌افزار با استفاده از روش‌های نیمه‌رسمی و رسمی انجام می‌شود که روش‌های رسمی به دلیل برخورداری از مفاهیم و مبانی ریاضی نسبت به روش‌های نیمه‌رسمی اعتمادپذیرتر هستند.

با توجه به روش‌های رسمی متعدد، گزینش یک روش مناسب برای توصیف و تحلیل یک نرم‌افزار خاص، یکی از چالش‌ها در مهندسی نرم‌افزار است. ارائه یک روش نظام‌مند که بتواند سطح توانایی روش‌های رسمی را برای توصیف و تحلیل نرم‌افزارهای مختلف تعیین کند، یک ضرورت می‌باشد. معمولاً روش‌های رسمی برای توصیف و تحلیل نرم‌افزارهای حساس به ایمنی[4]، با مأموریت خطیر[5]، نظامی، پزشکی، صنعتی و ابزار دقیق مورد استفاده قرار می‌گیرند [۳] و [۴]. اما توسعه‌دهندگان نرم‌افزار به دلایل زیر با مشکل انتخاب روش مناسب روبه‌رو هستند [۲]:

۱) تنوع و گستردگی روش‌های رسمی موجود.

۲) عدم آگاهی و دید کلی نسبت به مزایا و جایگاه مناسب هر روش در توسعه نرم‌افزار.

همچنین توانایی روش‌های رسمی در توصیف و تحلیل نرم‌افزار با توجه به معیارهای مختلف متفاوت است. برای مثال در حالی که شبکه‌های پتری[6] یک روش رسمی مناسب برای توصیف بصری نرم‌افزار و فهم آن است و همچنین دارای امکان توصیف همروندی و توزیع‌شدگی در نرم‌افزار است، تحلیل نرم‌افزار را نمی‌تواند بدون نگاشت به الگوهای ریاضی مانند جبر خطی انجام دهد یا مثال دیگر، زبان رسمی Z است که فاقد توصیف بصری نرم‌افزار و فاقد ابزار لازم برای نرم‌افزارهای مبتنی بر رخداد است اما به دلیل برخورداری از مفاهیم نظریه مجموعه‌ها و توابع مختلف ریاضی، قابلیت تحلیل ویژه‌ای در نرم‌افزارهای مبتنی بر حالت را داراست.

در بخش دوم به بررسی پیشینه تحقیق می‌پردازیم و تمایز کار صورت‌گرفته با تحقیق‌های مشابه را مورد کنکاش قرار می‌دهیم؛ در بخش سوم معیارهای سنجش توانایی روش‌های رسمی ارائه و این معیارها طبقه‌بندی می‌شوند؛ در بخش چهارم به معرفی و دسته‌بندی انواع نرم‌افزار و ویژگی‌های مورد نیاز برای توصیف آنها می‌پردازیم و از میان معیارهای ارائه‌شده در بخش سوم برای هر دسته از نرم‌افزار، معیار متناسبی را مشخص می‌کنیم. در بخش پنجم به معرفی و دسته‌بندی روش‌های مختلف رسمی می‌پردازیم و از هر دسته نمونه‌ای برای ارزیابی ارائه می‌کنیم که در انتهای این بخش روش‌های منتخب بر اساس معیارهای ارائه‌شده در بخش سوم، ارزیابی می‌شوند. در نهایت تعیین می‌کنیم که هر کدام از این روش‌ها برای توصیف کدام دسته از نرم‌افزارها مناسب‌تر هستند و در بخش ششم چند مورد مطالعه را مورد بررسی قرار می‌دهیم.

۲- کارهای مرتبط

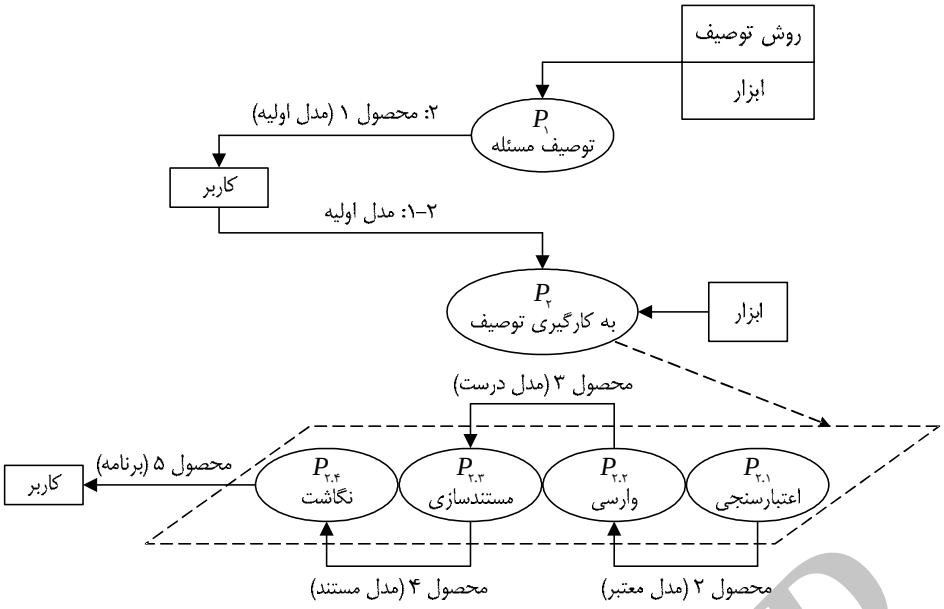
با توجه به تحقیق‌های انجام‌شده در مورد پیشینه تحقیق و کارهای صورت‌گرفته در این زمینه می‌توان دریافت که تاکنون رویکردی برای

این مقاله در تاریخ ۱۰ مهر ماه ۱۳۸۹ دریافت و در تاریخ ۲۰ دی ماه ۱۳۹۰ بازنگری شد. این تحقیق توسط دانشگاه کاشان بر اساس قرارداد پژوهشی شماره ۱۶۱۲۴ پشتیبانی شده است.

سیدمرتضی بابامیر، دانشکده مهندسی کامپیوتر، دانشگاه کاشان، کاشان، (email: babamir@kashanu.ac.ir).

ویدا احمدی ثابت، دانشگاه پیام نور مرکز همدان، همدان، (email: v.ahmadisabet@gmail.com).

1. Formal Methods
2. Semi - Formal
3. Unified Modeling Language
4. Safety - Critical
5. Mission - Critical
6. Petri Net



شکل ۱: نمودار زمینه برای سنجش روش‌های رسمی.

انتخاب یک روش رسمی مناسب از میان روش‌های رسمی موجود ارائه نشده است و تنها در برخی از مقاله‌ها که در ادامه به بررسی آنها می‌پردازیم مقایسه‌های موردی بین چند روش رسمی صورت گرفته است. البته در برخی از مقاله‌ها، معیارهایی جهت سنجش توانایی روش‌های رسمی ارائه شده اما در نهایت راهکاری جهت انتخاب یک روش رسمی، متناسب با نرم‌افزار تحت توسعه بیان نشده است.

مرجع [۵] توانایی‌ها و قابلیت‌های دو روش شبکه‌های پتری زمانی و اتوماتای زمان‌دار را از نظر قدرت بیان در توصیف نیازمندی‌ها مقایسه می‌کند. تنها نقطه قوت این مقاله تمرکز بر روی مقایسه دو روش رسمی است که از نمایش زمان پشتیبانی می‌نمایند و در این زمینه سعی نموده با انجام مقایسه‌هایی میان نمادهای مختلف موجود در این دو روش رسمی مشخص نماید که کدام روش دارای توانایی بیشتری می‌باشد. از نقاط ضعف این مقاله، عدم جامعیت آن و پوشش سایر معیارها و همچنین عدم پوشش و ارائه معیارهای مرتبط برای توصیف انواع نرم‌افزارها است. با توجه به این ضعف می‌توان نیاز به پوشش انواع نرم‌افزارهای مختلف را در این حیطه احساس نمود که در تحقیق جاری، گروه‌های مختلفی از نرم‌افزارها جهت جبران نقایص موجود در این مقاله پوشش داده شده است.

مرجع [۶] به معرفی روش‌های رسمی رایج و توصیف به‌وسیله آنها می‌پردازد و پس از دسته‌بندی روش‌های رسمی به چهار گروه اصلی (مانند روش‌های جبری و منطقی)، از هر دسته روش‌هایی را انتخاب می‌کند و توصیف به‌وسیله آن روش رسمی را شرح می‌دهد. در [۶]، ۱۲ معیار (از جمله همروندی، قابلیت اجرا، قابلیت اثبات) برای مقایسه روش‌های رسمی ارائه شده است اما علاوه بر این که برای معیارهای ارائه‌شده، طبقه‌بندی صورت نگرفته است، راهکاری برای انتخاب روش رسمی متناسب با نرم‌افزار نیز ارائه نشده است.

در [۷] که در سال ۲۰۰۷ ارائه شده است، یک مدل مرجع با استفاده از مقایسه و ترکیب روش‌های رسمی توسعه داده شده است. در این روش با استفاده از نتایج مقایسه روش‌های رسمی، نقص‌های هر روش تعیین می‌شود و سپس ترکیب روش‌ها پیشنهاد می‌شود. یکی از نقاط قوت این کتاب توسعه یک مدل مرجع با آگاهی از ضعف‌های روش‌های رسمی مختلف است اما در این کتاب نیز توجهی به طبقه‌بندی انواع نرم‌افزار و یافتن ارتباط بین معیارها و نرم‌افزارهای مختلف نشده است.

در [۸] که در سال ۲۰۱۱ ارائه شده، نویسندگان به بررسی وجوه تشابه و تمایز بین نمودارهای حالت UML و شبکه‌های پتری پرداخته‌اند. در این مقاله نیز تنها نمادهای این دو روش رسمی از نظر امکان توصیف مفاهیم توالی، همروندی، سلسله مراتب برای حالت‌ها و ارتباطات و همگامی مورد توجه قرار گرفته و معیارهایی جهت انجام مقایسه به‌صورت کلی بیان نشده است.

در [۹] که در سال ۲۰۰۲ ارائه شده، به بررسی توانایی شبکه‌های پتری، نمودارهای حالت، منطق زمانی، منطق زمانی بی‌درنگ و CSP برای توصیف نرم‌افزارهای بلادرنگ پرداخته شده است. اهمیت این مقاله از این منظر است که توانایی روش‌های رسمی مختلف را برای توصیف نرم‌افزارهای بلادرنگ مورد مقایسه قرار داده است، البته این مقاله نیز فقط به پوشش نرم‌افزارهای بلادرنگ می‌پردازد و راهکاری عمومی جهت انتخاب روش رسمی متناسب با نرم‌افزار ارائه نمی‌نماید.

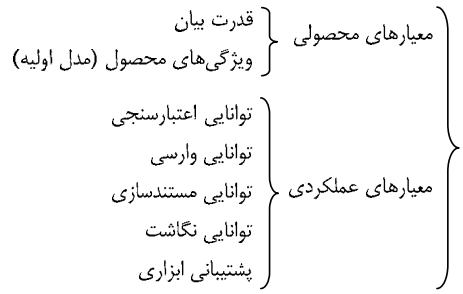
به‌طور کلی می‌توان مزایای این تحقیق را نسبت به کارهای مشابه در موارد زیر خلاصه نمود:

- ارائه معیارهای جامع‌تر جهت سنجش توانایی روش‌های رسمی از دیدگاه‌های مختلف مانند قابلیت نگاشت یا خوانایی (معیارهای ارائه‌شده در بخش سوم).
- طبقه‌بندی و گروه‌بندی معیارهای ارائه‌شده.
- برقرارکردن تطابق بین ویژگی‌های نرم‌افزارهای مختلف و معیارهای ارائه‌شده، جهت انتخاب روش رسمی متناسب با نوع نرم‌افزار.
- ارزیابی نمونه‌های منتخب از گروه‌های مختلف روش‌های رسمی.

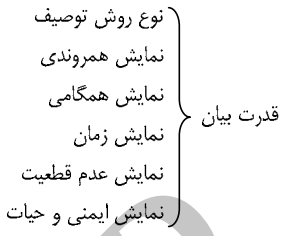
۳- معیارها

در این بخش ابتدا برای بیان مفاهیم کلی در ارتباط با روش‌های رسمی به ارائه نمودار زمینه که در شکل ۱ نشان داده شده است، می‌پردازیم. این نمودار به‌عنوان پایه و اساس کار جهت ساختاربندی معیارهایی که ارائه می‌کنیم، استفاده می‌گردد.

بر اساس شکل ۱ می‌توان گفت که ما یا در حال استفاده از روش‌های رسمی جهت توصیف مسئله هستیم که برای این بخش معیارهای محصولی را در نظر می‌گیریم و یا در حال به‌کارگیری توصیف‌ها جهت



شکل ۲: معیارهای سنجش روش‌های رسمی.



شکل ۳: قدرت بیان.

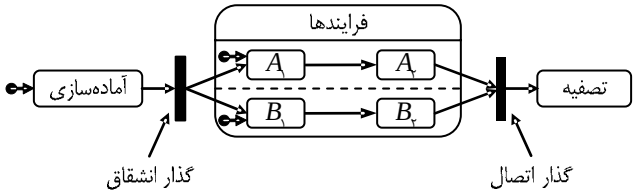
اعتبارسنجی، وارسی، مستندسازی یا نگاشت هستیم که برای این بخش نیز معیارهای عملکردی را لحاظ می‌نماییم. با توجه به این که از نحو و معنای روش رسمی جهت توصیف مسئله و ایجاد محصول ۱ (مدل اولیه) استفاده می‌شود، معیارهای محصولی را بر این اساس می‌توان به دو دسته قدرت بیان و ویژگی‌های محصول تقسیم نمود که این طبقه‌بندی در سطح بعدی در شکل ۲ نشان داده شده است.

۳-۱ معیارهای محصولی

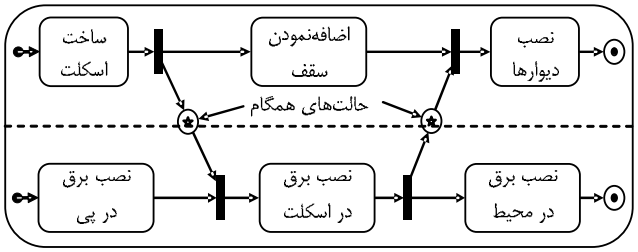
این گروه شامل معیارهایِ مرتبط با بخش تولید محصول است و بر اساس شکل ۱ شامل بند ۱ و ۲ (ویژگی‌های بیان و ویژگی‌های محصول ۱) است که با ارائه یک روش رسمی، یک نحو و یک دسته اصول معنایی برای آنها ارائه می‌گردد. نحوِ یک روش رسمی شامل مجموعه‌ای از نمادهای الفبایی تجزیه‌ناپذیر به همراه قواعدی برای تعریف و ساخت لغات به‌وسیله الفبای زبان می‌باشد. معانی این لغات به‌صورت ریاضی در بخش معنایی روش رسمی تفسیر می‌گردند که به‌طور کلی از نحو و معنای روش رسمی در این بخش جهت توصیف مسئله و ایجاد مدل اولیه یا همان محصول ۱ (بند ۲ در شکل ۱) استفاده می‌گردد. ما معیارهای محصولی را به قدرت بیان و ویژگی‌های محصول تقسیم می‌کنیم که در ادامه به شرح آنها می‌پردازیم.

الف) قدرت بیان (شکل ۳):

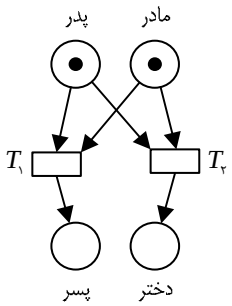
- نوع روش توصیف: این معیار تعیین‌کننده چگونگی نحوه استفاده از نمادهای روش، جهت توصیف نرم‌افزار می‌باشد و دارای یکی از مقادیر مبتنی بر حالت، مبتنی بر رخداد، جبری یا منطقی می‌تواند باشد که هر یک از این مقادیر در بخش پنجم به تفصیل مورد بررسی قرار خواهند گرفت.
- نمایش همروندی: این معیار به بررسی توانایی روش رسمی در توصیف فرایندهای موازی و همروند می‌پردازد. هنگامی که در یک نقطه از زمان بیش از یک فرایند در سیستم فعال باشد، گفته می‌شود که سیستم دارای فرایندهای همروند است. به‌طور مثال در نمودارهای حالت[1] برای نمایش همروندی از خطوط منقطع استفاده می‌شود که نمونه‌ای از همروندی در شکل ۴ نشان داده شده است.



شکل ۴: نمایش همروندی در نمودارهای حالت.



شکل ۵: نمایش همگامی در نمودارهای حالت.



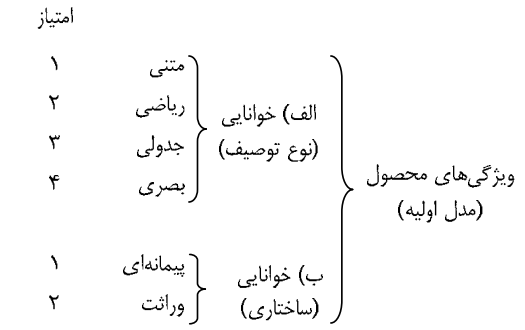
شکل ۶: توصیف شبکه پتری آماری برای ولادت فرزند.

- نمایش همگامی: در این معیار به بررسی توانایی روش رسمی در مدل‌کردن و نمایش همگامی مؤلفه‌های همروند پرداخته می‌شود. هنگامی که تعدادی فرایند در یک سیستم به‌صورت همروند صورت می‌گیرند، ممکن است زمان شروع این فرایندها به‌صورت هم‌زمان باشد که این مورد همگامی نامیده می‌شود. برای توصیف راه‌اندازی همگام تأسیسات برق، همراه با ساخت خانه، می‌توان به‌صورت شکل ۵ در نمودارهای حالت عمل نمود. این مفهوم در نمودارهای حالت با استفاده از حالت‌هایی که Synch State نامیده می‌شوند (که معمولاً با یک دایره به همراه یک ستاره در داخل آن نمایش داده شده و این دایره بین دو قسمت همروند قرار می‌گیرد) مدل می‌گردد.
- نمایش زمان: در این معیار، توانایی روش رسمی در بیان زمان مورد سنجش قرار می‌گیرد. عبارت $R(e,i,t)$ بیانگر iامین وقوع از رخداد e در زمان t با استفاده از منطق زمانی بی‌درنگ[2] می‌باشد. به‌طور مثال $R(\downarrow \text{SAMPLE}, ۱, x)$ بیانگر اولین وقوع کامل‌شدن کنش SAMPLE در زمان x می‌باشد.
- نمایش عدم قطعیت: هدف از این معیار بررسی امکان نمایش احتمال و آمار در مدل ارائه‌شده توسط روش رسمی می‌باشد. در روش‌هایی که از عدم قطعیت پشتیبانی می‌نمایند امکان نمایش گذارها یا کنش‌های احتمالی وجود دارد. در شکل ۶ شبکه پتری برای ولادت فرزند برای یک زوج ترسیم شده که در این شبکه پتری احتمال وقوع گذار T_1 و T_2، هر کدام ۵۰٪ است (رابطه (۱))

$$P(T_1) = ۰/۵ \quad , \quad P(T_2) = ۱ - ۰/۵ = ۰/۵ \qquad (۱)$$

1. Statecharts

2. Real-Time Logic



شکل ۷: ویژگی‌های محصول.

- نمایش ایمنی و حیات: این که در مدل ارائه‌شده اتفاق بدی نخواهد افتاد به امن‌بودن مدل تعبیر می‌شود. به‌طور مثال برای امن‌بودن سیستم تولیدکننده، مصرف‌کننده باید مصرف‌کننده هر پیام را قبل از این که تولیدکننده پیام دیگری را تولید نماید، از حافظه بردارد. برای بیان این مطلب می‌توان از رابطه منطقی (۲) در منطق زمانی بی‌درنگ بهره برد. در این فرمول منطقی، بیان شده است که عملیات شروع مصرف $(\uparrow Q)$ نمونه iام توسط مصرف‌کننده باید قبل از تولید نمونه $(i+1)$ام توسط تولیدکننده صورت گیرد

$$\forall i : @(\uparrow Q, i) \le @(P, i+1) \qquad (۲)$$

مفهوم حیات بدین معناست که در مدل ارائه‌شده برای سیستم، بالاخره اتفاق خوبی خواهد افتاد. به‌طور مثال برای بیان گزاره "موقعی که ریل خالی است، مانع باید بالا باشد تا اجازه عبور داده شود" می‌توان به (۳) در روش ASM[1] اشاره کرد. رابطه (۳) اظهار می‌دارد در صورتی که قطاری در حال عبور از خط نباشد، باید زمانی بین α و β باشد که در آن زمان بسته‌بودن گیت (d_{close}) برابر حداکثر زمان دیده‌شدن $(d_{\max})$ قطار منهای زمان انتظار برای بسته‌شدن گیت باشد (WaitTime)

$$\begin{aligned} &\text{for every } x, \text{TrackStatus}(x) \neq inCrossing \\ &\text{interval}(\alpha, \beta) \text{ with } \alpha + d_{\text{open}} < \beta - d_{\text{close}} \\ &d_{\text{close}} = d_{\max} - WaitTime = d_{\text{close}} + (d_{\max} - d_{\min}) \end{aligned} \qquad (۳)$$

ب) ویژگی‌های محصول (شکل ۷):

- خوانایی (نوع توصیف): خوانایی یک روش رسمی را بر اساس نوع الفبای آن جهت توصیف یک نرم‌افزار به‌صورت زیر دسته‌بندی و امتیازدهی می‌نماییم. به‌طور کلی می‌توان گفت روش‌های بصری دارای بیشترین خوانایی و روش‌های متنی دارای کمترین خوانایی می‌باشند.
  - متنی: استفاده از الفبای زبان محاوره‌ای جهت توصیف نرم‌افزار با امتیاز ۱.
  - ریاضی: استفاده از نمادهای ریاضی مانند سورهای عمومی و وجودی ($\forall$ و $\exists$)، $\sum$ که در زبان محاوره استفاده نمی‌شوند با امتیاز ۲.
  - جدولی: استفاده از جداول جهت توصیف نرم‌افزار با امتیاز ۳.
  - بصری: استفاده از علایم بصری جهت توصیف نرم‌افزار با امتیاز ۴.
- خوانایی (ساختاری): خوانایی یک روش رسمی را بر اساس نوع پیمانه‌سازی روش مشخص می‌کنیم. روش‌های رسمی پیمانه‌سازی صرف با خوانایی کمتر و روش‌های رسمی پیمانه‌سازی با امکان پشتیبانی از وراثت با خوانایی بیشتر می‌باشند.

1. Abstarct State Machine

شکل ۸: توانایی اعتبارسنجی.

- پیمانه‌سازی: در صورتی که یک روش رسمی برای توصیف از پیمانه‌ها استفاده نماید، توصیف کلی یک نرم‌افزار بزرگ و پیچیده را از طریق تجزیه نرم‌افزار به بخش‌های کوچک انجام می‌دهد که توصیف این بخش‌ها در قالب پیمانه‌ها صورت می‌گیرند. به چنین روش‌های رسمی امتیاز ۱ را نسبت می‌دهیم.
- وراثت: در این معیار به بررسی نمادهای نحوی یک روش رسمی جهت پشتیبانی از مفاهیم وراثت می‌پردازیم. روش‌های رسمی که از وراثت پشتیبانی می‌نمایند به آنها امتیاز ۲ داده می‌شود.

۳-۲ معیارهای عملکردی

این گروه دربرگیرنده تمام فعالیت‌هایی است که در زمینه استفاده از توصیف‌های رسمی صورت می‌گیرد و بر طبق شکل ۱ شامل اعتبارسنجی، وارسی، مستندسازی، نگاشت و پشتیبانی ابزاری می‌باشد. در ادامه به بررسی هر دسته از این معیارها می‌پردازیم.

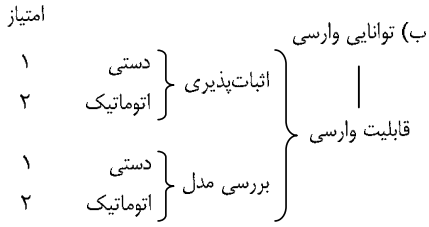
الف) توانایی اعتبارسنجی: هنگامی که توصیف‌های رسمی از یک نرم‌افزار در دست است، مسئله بررسی صحت توصیف‌های ارائه‌شده از اهمیت قابل توجهی برخوردار است که این بررسی را اعتبارسنجی گویند. دسته‌بندی این معیار در شکل ۸ نشان داده شده است.

- قابلیت اجرا: اجرای توصیف‌های رسمی یک نرم‌افزار، یکی از راه‌های بررسی صحت و اعتبارسنجی توصیف‌های ارائه‌شده می‌باشد. برای اجرای توصیف‌های یک نرم‌افزار، می‌توان از موقعیت‌های مختلف توصیف، شروع و با فرض وقوع رخدادهای مرتبط با نرم‌افزار موقعیت‌های بعدی، اجرای نرم‌افزار را دنبال نمود و ضمناً باید به بررسی صحت موقعیت فعلی و تطابق آن با توجه به ویژگی‌ها و رفتار مورد انتظار پرداخت. بنابراین در صورتی که بتوان توصیف‌های ارائه‌شده را اجرا نمود تا شبیه‌سازی از رفتار سیستم در دست باشد، این ویژگی قابلیت اجرا گفته می‌شود. به قابلیت اجرای توصیف‌ها به صورت دستی امتیاز ۱ و به قابلیت اجرا به صورت اتوماتیک (پشتیبانی ابزاری) امتیاز ۲ را نسبت می‌دهیم.

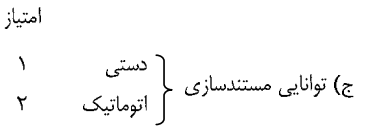
ب) توانایی وارسی:

- قابلیت وارسی: فرایند بررسی و تحلیل یک سیستم جهت داشتن ویژگی‌های مطلوب، وارسی نامیده می‌شود. دو خط مشی ساخته‌شده برای وارسی توصیف‌های رسمی شامل اثبات قضیه[2] و بررسی مدل[3] می‌باشد (شکل ۹).
- توانایی اثبات: اثبات قضیه، تکنیکی است که به‌وسیله آن ویژگی‌های مطلوب نرم‌افزار از طریق فرمول ریاضی منطقی اثبات می‌شوند و منطق ارائه‌شده در این تکنیک شامل مجموعه‌ای از اصول[4] و مجموعه‌ای از قوانین استنتاجی است. به‌طور خلاصه می‌توان گفت اثبات قضیه، فرایند یافتن یک اثبات برای یک ویژگی با استفاده از اصول می‌باشد. در این معیار در صورت اثبات به‌صورت دستی امتیاز ۱ و در صورت اثبات به‌صورت خودکار به آن امتیاز ۲ را می‌دهیم.

2. Theorem Proving
3. Model Checking
4. Axiom

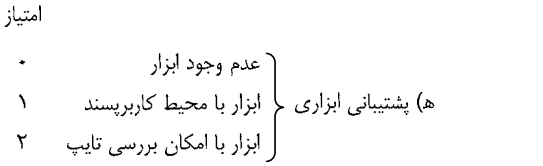


شکل ۹: توانایی وارسی.

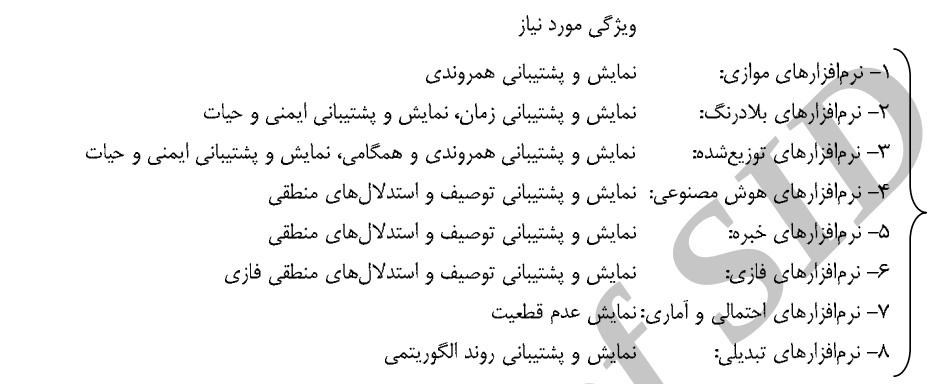


شکل ۱۰: توانایی مستندسازی.

شکل ۱۱: توانایی نگاشت.



شکل ۱۲: پشتیبانی ابزاری.



شکل ۱۳: دسته‌بندی انواع نرم‌افزار و تعیین ویژگی‌های مورد نیاز آنها.

- بررسی مدل: بررسی مدل، روشی است که بر ساخت مدل متناهی از حالت‌های نرم‌افزار و بررسی وجود ویژگی‌های مطلوب در میان این حالت‌ها تکیه دارد. این بررسی به‌صورت جامع فضای حالت یک نرم‌افزار را جستجو کرده و تضمین می‌نماید که مدل یک سیستم، محدود، خاتمه‌پذیر و دارای ویژگی مورد نظر است. در این معیار در صورت امکان بررسی مدل به صورت دستی امتیاز ۱ و در صورت امکان بررسی مدل به صورت خودکار امتیاز ۲ را می‌دهیم.

ج) توانایی مستندسازی: توصیف‌های ارائه‌شده توسط برخی از روش‌های رسمی می‌تواند به‌عنوان مستندات یک نرم‌افزار (به منزله انقیاد قراردادی مابین مشتری و طراح نرم‌افزار) استفاده شوند. در این معیار به بررسی توانایی مستندسازی روش رسمی برای توصیف‌های ارائه‌شده می‌پردازیم و در صورت امکان مستندسازی به صورت دستی امتیاز ۱ و در صورت امکان مستندسازی به صورت خودکار امتیاز ۲ را می‌دهیم (شکل ۱۰).

د) توانایی نگاشت: در صورتی که روش رسمی مد نظر دارای قابلیت نگاشت باشد، به راحتی می‌توان توصیف‌های صورت‌گرفته را به یک یا چند زبان برنامه‌نویسی تبدیل نمود که داشتن چنین ویژگی موجب می‌گردد تا توسعه نرم‌افزار هم سریع‌تر و هم با تطابق بیشتر با توصیف‌ها صورت گیرد. در صورت امکان نگاشت به‌صورت دستی امتیاز ۱ و در صورت امکان نگاشت به‌صورت اتوماتیک امتیاز ۲ را می‌دهیم (شکل ۱۱).

ه) پشتیبانی ابزاری: داشتن ابزار برای یک روش رسمی یکی از معیارهای مهم تلقی می‌گردد زیرا با استفاده از یک ابزار که دارای یک محیط کاربرپسند بوده و امکان بررسی دستور زبان روش‌های رسمی را فراهم می‌نماید، فرایند توصیف نرم‌افزار راحت‌تر و با دقت بیشتری امکان‌پذیر می‌گردد. در صورت عدم وجود ابزار، امتیاز صفر و در صورت وجود ابزار با یک محیط کاربرپسند بدون داشتن امکان بررسی دستور زبان، امتیاز ۱ و در صورت وجود ابزار برای یک روش رسمی با امکان بررسی دستور زبان توصیف‌ها، امتیاز ۲ را به روش رسمی می‌دهیم (شکل ۱۲).

٤- طبقه‌بندی نرم‌افزارها

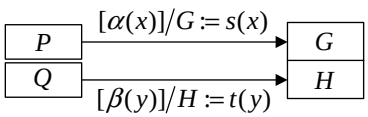
در این بخش به طبقه‌بندی نرم‌افزارها و ویژگی‌های هر طبقه می‌پردازیم. به‌طور کلی انواع نرم‌افزار را به هشت دسته کلی تقسیم می‌نماییم و برای هر دسته از نرم‌افزار، ویژگی مورد نیاز را بر اساس نوع نرم‌افزار تعیین می‌کنیم (شکل ۱۳).

٤-۱ نرم‌افزارهای موازی در مقابل نرم‌افزارهای ترتیبی

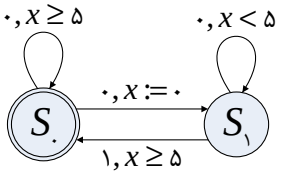
به‌طورکلی نرم‌افزارها می‌توانند به‌صورت موازی یا ترتیبی باشند. در یک نرم‌افزار موازی، دو فرایند در سیستم با یکدیگر در تعامل بوده و در یک نقطه از زمان با هم اجرا می‌شوند [۱۰]. در صورت موازی انجام شدن، فرایندهای موازی می‌توانند همگام یا ناهمگام باشند. ویژگی‌های مورد نیاز برای این دسته از نرم‌افزارها عبارتند از:

- همروندی: با توجه به تعریفی که برای این دسته از نرم‌افزارها ارائه شد، برخی از فرایندها در این گونه نرم‌افزارها به‌صورت موازی اجرا می‌شوند که برای مدل‌کردن و توصیف این فرایندها لازم است که روش رسمی انتخاب شود که امکان نمایش همروندی را دارا باشد. در روش رسمی CSP[1] برای نمایش همروندی دو فرایند P که دارای دو کنش a و b و فرایند Q که دارای دو کنش c و d می‌باشد از عبارت $P \| Q$ استفاده می‌شود که توصیف رسمی این

1. Communicating Sequential Processes



شکل ۱۴: نمایش هم‌گامی برای دو فرآیند P و Q در مدل پایه ASM.



شکل ۱۵: نمایش زمان با استفاده از اتوماتای زمان‌دار.

عبارت در (۴) بیان شده است

$$P\|Q = P[\![Q + Q[\![P + P|Q = (a.b)[\![(c.d) + (c.d)[\![(a.b) + (a.b)|(c.d) \quad (۴)$$

رابطه (۴) اظهار می‌دارد که کنش‌های فرایند P می‌توانند زودتر از Q انجام شوند ($P[\![Q$) یا کنش‌های فرایند Q می‌تواند زودتر از P انجام پذیرند و یا کنش‌های این دو فرایند در حین اجرا با یکدیگر در تعامل هستند ($P|Q$). طرف دوم (۴) با جایگذاری کنش‌های دو فرایند P و Q حاصل می‌شود. برای نمونه $(a.b)[\![(c.d)$ بیانگر این است که یکی از دنباله‌های $abcd$، $acdb$ یا $acbd$ اتفاق می‌افتند و تضمین می‌نماید a از فرایند P زودتر از سایر کنش‌ها اتفاق بیافتد.

- همگامی: برخی از فرایندهای همروند، همگام و برخی غیر همگام می‌باشند. در صورت همگام‌بودن فرایندهای موازی لازم است تا معیار نمایش همگامی نیز مورد توجه قرار گیرد. در شکل ۱۴ دو فرایند P و Q نشان داده شده که این دو فرایند قرار است به‌صورت همگام، مقادیر G و H را به‌ترتیب با مقادیر s و t به‌روز رسانی نمایند. پیش‌شرط این به‌روز رسانی برای فرایند P این است که x دارای مقدار α بوده و پیش‌شرط به‌روز رسانی برای فرایند Q این است که y دارای مقدار β باشد.

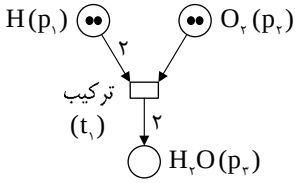
توصیف رسمی برای شکل ۱۴ در (۵) بیان شده است

$$\begin{aligned}&\text{AspHandshaking}(P,\alpha,s,Q,\beta,t) = \text{sync}(G,H)\{R,S\}\\&\text{where}\\&R = \text{choose } x \text{ with } \alpha(x) \text{ in } (G := s(x) \text{ then } P(x))\\&S = \text{choose } y \text{ with } \beta(y) \text{ in } (H := t(y) \text{ then } Q(y))\end{aligned} \quad (۵)$$

در گزاره اول (۵) دو فرایند P و Q و دو رخداد مرتبط با آنها یعنی R و S معرفی شده‌اند و با نماد sync بر همگامی این دو فرایند اذعان شده است. در گزاره دوم با استفاده از قاعده choose پیش‌شرط اجرای رخداد R، دارابودن مقدار α برای متغیر x مشخص شده و کنش این رخداد، مقداردهی s به فضای G می‌باشد. در گزاره سوم نیز پیش‌شرط اجرای رخداد S دارابودن مقدار β برای متغیر y مشخص شده و کنش این رخداد، مقداردهی t به فضای H می‌باشد.

۲-٤ نرم‌افزارهای بلادرنگ/ واکنشی

یک سیستم بلادرنگ عبارت است از سیستمی که مسأله زمان برای تولید خروجی فاکتور مهمی بوده و این بدان جهت است که ورودی این سیستم بر طبق تغییرها، در محیط وارد شده و خروجی مرتبط با این تغییرها تولید می‌گردد. تأخیر زمانی بین ورودی و خروجی در این سیستم‌ها



شکل ۱۶: نمایش واکنش در توصیف تشکیل آب با استفاده از شبکه‌های پتری.

می‌بایست تا حد مناسب تعریف‌شده برای این دسته از سیستم‌ها کوچک باشد. نرم‌افزارهای بلادرنگ گاهاً نرم‌افزارهای واکنشی نیز نامیده می‌شوند زیرا به‌صورت مداوم در تعامل با محیط بوده و بر اساس رخدادها در محیط پاسخ مناسب ایجاد می‌نمایند. ویژگی‌های مورد نیاز برای این نرم‌افزارها عبارتند از:

- زمان: نمایش زمان در این نرم‌افزارها مهم‌ترین عامل بوده و یک روش رسمی لازم است که بتواند زمان را در مدل ارائه‌شده نمایش دهد. به‌طور مثال می‌خواهیم ماشینی با استفاده از اتوماتای زمان‌دار[1] مدل کنیم که این ماشین دارای الفبای $\Sigma = \{۰,۱\}$ بوده و زبان مورد قبول برای این ماشین، دنباله‌ای از صفرها در زمان کمتر از ۵ بوده و در زمان مساوی یا بیشتر از ۵، یک عدد یک پذیرش و بعد از آن تعداد دلخواهی صفر است (شکل ۱۵).

تعریف رسمی اتوماتای شکل ۱۵ را با (۶) نشان داده‌ایم که L زبان مورد پذیرش ماشین باشد. زوج (σ,τ) کلمه زمان‌دار[2] روی الفبای Σ است که در آن σ یک کلمه نامتناهی روی Σ و τ بیانگر دنباله افزایشی زمان است. در خط اول (۶)، دنباله‌ای از صفرها قبل از زمان ۵ پذیرش می‌شوند. در خط دوم (۶)، حداقل یک صفر در زمان کمتر از ۵ و مقدار یک در زمان ۵ یا بیشتر پذیرش می‌شود و در خط سوم (۶)، پس از پذیرش مقدار یک، دنباله‌ای از صفرها پذیرش می‌شود

$$\begin{aligned}L = \{(\sigma,\tau) \mid &\forall i : (\tau_i < ۵ \rightarrow \sigma_i := ۰) \wedge\\ &\exists j : ((\tau_j \geq ۵) \wedge \forall i : (\tau_i < \tau_j \wedge \sigma_i := ۰)) \rightarrow\\ &(\sigma_j := ۱) \wedge \forall i : ((\tau_i \geq ۵) \wedge \exists j : (j < i \wedge \sigma_j = ۱) > \sigma_i := ۰)\end{aligned} \quad (۶)$$

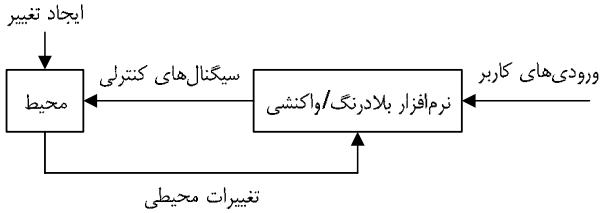
- واکنش: ویژگی بعدی مورد توجه در این گونه نرم‌افزارها واکنش در مقابل رخدادها و تغییرهای محیطی می‌باشد که برای مدل‌کردن این ویژگی نیز باید روش رسمی انتخاب نمود که دارای امکان نمایش حیات و ایمنی باشد (به مفاهیم ایمنی و حیات در بخش سوم رجوع شود). برای نمونه شبکه پتری را برای تشکیل دو مولکول آب ترسیم می‌نماییم که در این شبکه پتری مراحل واکنش یک مولکول اکسیژن با دو مولکول هیدروژن نشان داده شده است (شکل ۱۶).

توصیف رسمی واکنش برای شکل ۱۶ در (۷) بیان شده است

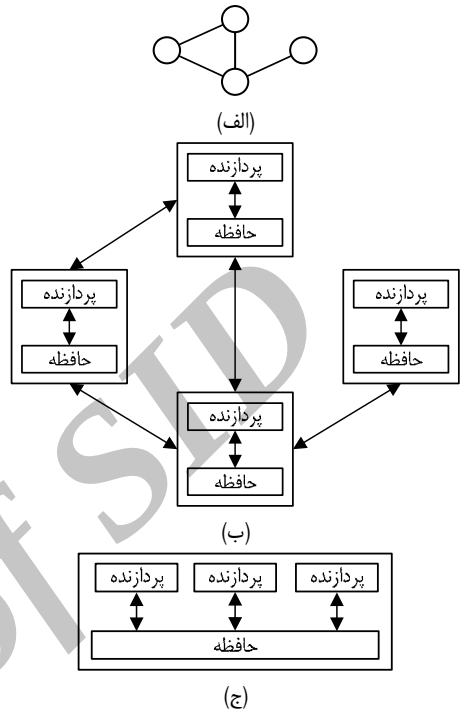
$$\begin{aligned}&(S,T,W,M_۰) \geq S = (p_۱, p_۲, p_۳)\\&\text{where}\\&T = (t_۱)\ ,\ W = \{(p_۱,t_۱),(t_۱,p_۳),(p_۲,t_۱)\}\\&M_۰ = (۲,۲,۰) \xrightarrow{t_۱} M_۱(۰,۱,۲)\end{aligned} \quad (۷)$$

در (۷) شبکه پتری به‌صورت یک چندتایی شامل S (لیست مکان‌ها)، T (لیست گذارها)، W (بیانگر کمان‌ها) و $M_۰$ (حالت اولیه شبکه) بیان شده است. در این رابطه $M_۰$ اظهار می‌دارد که با

1. Timed - Automata
2. Timed Word



شکل ۱۷: ساختار نرم‌افزارهای بلادرنگ/ واکنشی [۹].



شکل ۱۸: (الف) و (ب) یک سیستم توزیع‌شده و (ج) یک سیستم موازی.

آتش‌شدن گذار t_1، دو نشانه از مکان p_1 و یک نشانه از مکان p_2 جدا و دو نشانه به مکان p_3 اضافه می‌شود. شکل ۱۷ ساختار نرم‌افزارهای بلادرنگ را نشان می‌دهد [۹].

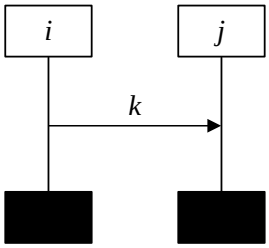
۴-۳ نرم‌افزارهای توزیع‌شده

اساس کار نرم‌افزارهای توزیع‌شده بر این است که پردازش بین چند کامپیوتر مستقل که هر کدام دارای حافظه مستقل برای خود هستند و به‌صورت فیزیکی از لحاظ جغرافیایی توزیع‌شده می‌باشند، صورت می‌گیرد. ارتباط بین این کامپیوترها معمولاً از طریق سیستم پیام‌رسانی صورت می‌گیرد. در یک سیستم موازی، همه پردازنده‌ها به یک حافظه مشترک دسترسی داشته (شکل ۱۸- ج) و این حافظه مشترک جهت تبادل اطلاعات بین پردازنده‌ها استفاده می‌شود، اما در سیستم‌های توزیع‌شده هر پردازنده دارای حافظه مستقل برای خود بوده و تبادل اطلاعات به‌وسیله پیام‌رسانی صورت می‌گیرد [۱۱] (شکل ۱۸- الف و ۱۸- ب).

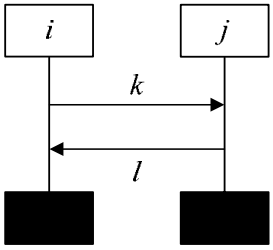
بر اساس مشخصات ذکرشده برای نرم‌افزارهای توزیع‌شده و بر اساس [۱۲] قابلیت‌هایی که یک روش رسمی جهت توصیف و تحلیل این نرم‌افزارها می‌بایست دارا باشد، توزیع‌شدگی، پیام‌رسانی و مدل‌سازی ویژگی‌های ایمنی و حیات است. در ادامه به شرح این قابلیت‌ها می‌پردازیم.

- توزیع‌شدگی: در نرم‌افزارهای توزیعی، پردازش‌ها بین چند کامپیوتر مستقل توزیع می‌شود. شکل ۱۹، توزیع‌شدگی بین دو کامپیوتر مستقل را با استفاده از روش MSC[1] نشان می‌دهد. توصیف رسمی

1. Message Sequence Chart



شکل ۱۹: نمایش دو نمونه توزیع‌شده با استفاده از MSC.



شکل ۲۰: نمایش پیام‌رسانی در MSC.

برای شکل ۱۹ در (۸) بیان شده است

$$\begin{aligned}&\lambda(out(i,j,k)\,\|\,in(i,j,k))\\&=\lambda(out(i,j,k).in(i,j,k)+in(i,j,k).out(i,j,k))\\&=out(i,j,k).in(i,j,k)\end{aligned}\quad(۸)$$

رابطه (۸) اظهار می‌دارد که نمونه i ام پیام k را برای نمونه j ام جهت پردازش ارسال می‌کند و به‌صورت موازی نمونه j ام پیام k را جهت پردازش دریافت می‌نماید. نماد λ در این رابطه بیانگر این است که پیام k حتماً قبل از رسیدن می‌بایست توسط نمونه i ام ارسال شده باشد. خط دوم (۸) اظهار می‌دارد که پیام k ابتدا ارسال و سپس توسط نمونه j دریافت می‌شود یا ابتدا پیام توسط j دریافت و سپس توسط نمونه i ارسال می‌گردد. اما با توجه به نماد λ می‌توان نتیجه گرفت که فقط مورد اول امکان‌پذیر می‌باشد که این نتیجه در خط سوم (۸) بیان شده است.

- پیام‌رسانی: با توجه به این که لازم است توصیف ارتباطات بین فرایندها در این گونه نرم‌افزارها صورت گیرد، روش رسمی انتخابی باید دارای امکان نمایش همروندی و همگامی بین فرایندها باشد. شکل ۲۰ پیام‌رسانی بین دو نمونه توزیع‌شده را در روش MSC نشان می‌دهد.

توصیف رسمی برای شکل ۲۰ در (۹) بیان شده است

$$\begin{aligned}&\lambda(out(i,j,k).in(j,i,l)\,\|\,in(i,j,k).out(j,i,l))\\&=out(i,j,k).(in(i,j,k).(in(j,i,1).out(j,i,1)\\&+out(j,i,1).in(j,i,1))\\&+in(j,i,1).in(i,j,k).out(j,i,1))\\&+in(i,j,k).(out(i,j,k).(out(j,i,1).in(j,i,1)\\&+in(j,i,1).out(j,i,1))\\&+out(j,i,1).out(i,j,k).in(j,i,1))\\&=out(i,j,k).in(i,j,k).out(j,i,1).in(j,i,1)\end{aligned}\quad(۹)$$

رابطه (۹) اظهار می‌دارد که پیام k از نمونه i ام به نمونه j ام ارسال و پیام l توسط نمونه i ام از نمونه j ام دریافت می‌شود و به‌طور موازی نمونه j ام پیام l را برای نمونه i ام ارسال و پیام k را از نمونه i ام دریافت می‌دارد. خط آخر (۹) پس از حذف تمامی حالت‌های ناممکن به‌دست می‌آید.

- مدل‌کردن ایمنی و حیات: در سیستم‌های توزیع‌شده لازم است تا ایمنی و حیات نیز در مدل ارائه‌شده قابل نمایش باشد.

٤-٤ نرم‌افزارهای هوش مصنوعی

این دسته از نرم‌افزارها بر اساس تعاریف موجود، ادراک‌هایی از محیط داشته و بر اساس آن واکنش‌هایی مناسب ارائه می‌دهند. نرم‌افزار هوش مصنوعی در تلاش است تا واکنش ارائه‌شده، مناسب‌ترین پاسخ به رخداد موجود در محیط باشد. این گونه نرم‌افزارها دارای ویژگی‌های استدلال، دانش[1]، برنامه‌ریزی، یادگیری، ارتباط، ادراک[2] و توانایی حرکت و جابه‌جایی اشیا می‌باشند [۱۳]. مسئله اساسی در تولید یا شبیه‌سازی نرم‌افزارهای هوش مصنوعی مسئله دانش و استدلال بر پایه دانش است. بر اساس این دو ویژگی مهم می‌توان گفت که برای پوشش مبحث دانش، نرم‌افزار باید بتواند اشیا، ویژگی‌های اشیا، دسته‌های آنها، ارتباط بین اشیا، زمان، علل و تأثیرات آن را در بر داشته باشد و این یک نمایش کامل از چیزهایی است که وجود دارد و این نمایش آنتولوژی[3] نامیده می‌شود. آنتولوژی در علوم کامپیوتر یک نمایش رسمی از مفاهیم یک حوزه و ارتباط بین آنها است. این مفاهیم جهت توصیف حوزه و استدلال درباره حوزه خاص مورد استفاده قرار می‌گیرند [۱۴]. به‌صورت تئوری آنتولوژی یک توصیف صریح رسمی از مفاهیم بوده که خود شامل یک فرهنگ لغت برای مدل‌کردن یک حوزه خاص می‌باشد. از جمله زبان‌های رسمی برای توصیف آنتولوژی می‌توان به DOGMA[4] [۱۵] و OBO[5] [۱۶] اشاره نمود.

اما دومین ویژگی اساسی این گونه نرم‌افزارها، استدلال بر پایه دانش است. نتایج تحقیق‌های متعدد John McCarthy [۱۷] در آزمایشگاهش در استنفورد نشان داده است که استفاده از روش‌های رسمی منطقی برای این گونه نرم‌افزارها بهترین راه می‌باشد. تلاش‌های وی و تحقیق‌های صورت‌گرفته در دانشگاه Edinburgh و سایر تحقیق‌های مرتبط منجر به ایجاد زبان برنامه‌نویسی Prolog و دانش برنامه‌نویسی منطقی[6] شده است [۱۸]. به‌طور کلی برای یک توسعه‌دهنده که در صدد توسعه یک نرم‌افزار هوش مصنوعی است باید ویژگی‌های زیر را لحاظ کرد:

- ویژگی استدلال: با توجه به نیاز به استدلال در این گونه نرم‌افزارها می‌توان جهت توصیف از روش‌های رسمی منطقی استفاده نمود و از همین رو باید معیار نوع روش توصیف مورد توجه قرار گیرد. رابطه (۱۰) نمونه‌ای از استدلال استنتاجی را نشان می‌دهد که در آن فرض اول اظهار می‌دارد که اگر نوشیدنی از آب جوش تهیه شده است، داغ است و فرض دوم اظهار می‌دارد که این نوشیدنی از آب جوش تهیه نشده است. نتیجه این دو فرض این است که این نوشیدنی داغ نیست

Premise ۱: If a drink is made with boiling water, it will be hot.
Premise ۲: This drink was not made with boiling water. (۱۰)
Conclusion: This drink is not hot.

٤-٥ نرم‌افزارهای خبره

یک نرم‌افزار خبره نرم‌افزاری است که در تلاش برای فراهم‌نمودن پاسخ برای یک مسئله بوده یا در مواردی که به‌طور طبیعی دو یا چند فرد خبره نیاز به مشورت و هم‌فکری دارند، شبهات را برطرف می‌نماید. سیستم‌های خبره معمولاً برای یک حوزه خاص از مسائل مورد استفاده هستند. این گونه نرم‌افزارها دارای یک پایگاه دانش بوده که اطلاعات این پایگاه دانش از افراد خبره جمع‌آوری می‌شود و در واقع این گونه سیستم‌ها نسل قدیمی و اولین پیاده‌سازی موفقیت‌آمیز از دسته نرم‌افزارهای هوش مصنوعی می‌باشند [۱۹]. ویژگی‌های این نرم‌افزارها ضریب اطمینان و استدلال است که در ادامه به شرح آنها می‌پردازیم.

- ضریب اطمینان: در یک استدلال ممکن است که از نتیجه‌گیری خود به‌طور قطعی مطمئن نباشیم. به‌طور مثال می‌توان گفت اگر حیوانی سبز است، شاید آن حیوان قورباغه باشد (در حالی که ممکن است سوسمار هم باشد). این نوع از استدلال با استفاده از ضریب اطمینان صورت می‌گیرد و به این صورت بیان می‌شود که اگر حیوانی سبز است با اطمینان ۸۵٪ قورباغه است. این عبارت به‌صورت (۱۱) نمایش داده می‌شود

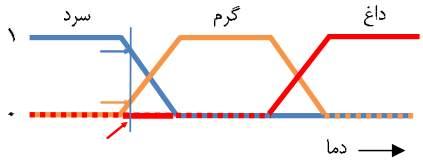
$$\text{if } X \text{ is } green \text{ then } X \text{ is } a\ frog.(CF:+۸۵\%) \qquad (۱۱)$$

- استدلال: بعد دیگر راجع به سیستم‌های خبره زنجیره یا تسلسل[7] می‌باشد. دو روش زنجیره رو به جلو و زنجیره رو به عقب برای استدلال با استفاده از قواعد استنتاجی وجود دارد. زنجیره رو به جلو از داده‌های موجود و با استفاده از قواعد استنتاجی شروع شده و برای نتیجه‌گیری داده‌های بیشتر تا رسیدن به هدف ادامه می‌یابد. اما زنجیره رو به عقب از اهداف شروع‌شده به‌صورت خلفی تا رسیدن به داده‌ها ادامه می‌یابد. به‌طور مثال فرض کنید که قوانین پایه شامل (۱۲) باشد و هدف ما حیوانی است که می‌جهد

$$\begin{aligned}
&\text{if } X \text{ is } green\\
&\quad \text{then } X \text{ is } a\ frog.(CF:+۱\%)\\
&\text{if } X \text{ is } green\\
&\quad \text{then } X \text{ is not } a\ frog.(CF:+۹۹\%)\\
&\text{if } X \text{ is } a\ frog\\
&\quad \text{then } X\ hops.(CF:+۵۰\%)\\
&\text{if } X \text{ is } a\ frog\\
&\quad \text{then } X \text{ does not } hop.(CF:+۵۰\%)
\end{aligned} \qquad (۱۲)$$

حال با استفاده از روش استدلالی زنجیره رو به عقب، قانون ۳ را به کار می‌بریم زیرا نتیجه این قانون با هدف برابر است. از قانون ۳ به این نتیجه می‌رسیم که این حیوان می‌تواند با ضریب اطمینان ۵۰٪ قورباغه باشد که این نتیجه به اهداف ما اضافه می‌شود و دوباره قوانین جستجو می‌شوند. قاعده ۱ که با هدف ما برابر است انتخاب می‌شود و از این قاعده به این نتیجه می‌رسیم که آن حیوان با اطمینان ۱٪ سبز رنگ است. پس از کسب این نتیجه و ضرب ضرایب اطمینان، می‌توان گفت حیوانی که می‌جهد با اطمینان ۰/۰۰۵ سبز رنگ است. در این گونه سیستم‌ها هرچه پایگاه دانش کامل‌تر باشد، نتیجه‌گیری دقیق‌تر خواهد بود. سیستم خبره رهن یک نمونه مناسب از این گونه سیستم‌ها می‌باشد که با سؤالات

1. Knowledge
2. Perception
3. Ontology
4. Developing Ontology-Grounded Methods and Applications
5. Open Biomedical Ontologies
6. Logic Programming
7. Chaining



شکل ۲۱: نمودار منطق فازی برای دما.

متعدد آپارتمان مورد نظر مشتری را از پایگاه داده خود استخراج می‌نماید. با توجه به مطالب بیان‌شده و ویژگی‌های سیستم‌های خبره می‌توان گفت برای توصیف این گونه سیستم‌ها نیاز به روش رسمی است که از منطق شرطی رتبه اول (توجه به معیار نوع روش توصیف) پشتیبانی نماید.

٤-٦ نرم‌افزارهای فازی

یک نرم‌افزار فازی، یک سیستم نرم‌افزاری بر پایه منطق فازی می‌باشد. منطق فازی که به‌طور گسترده در کنترل سیستم استفاده می‌شود به‌صورت یک منطق چندمقداری[1] نشان داده شده و از تئوری مجموعه فازی استخراج شده است. ویژگی این گونه نرم‌افزارها، فازی‌بودن است که در ادامه به شرح آن می‌پردازیم.

- فازی‌بودن: بر خلاف منطق دودویی که فقط شامل مقادیر گسسته صفر یا ۱ می‌باشد، متغیرهای منطق فازی می‌توانند در محدوده صفر تا ۱ دارای مقدار باشند. منطق فازی در ادامه تئوری مجموعه فازی در سال ۱۹۶۵ توسط لطفی‌زاده [۲۰] پیشنهاد شده است و در بسیاری از زمینه‌ها از جمله تئوری کنترل (یک شاخه از علم مهندسی و ریاضی که به بررسی رفتار پویای سیستم‌ها می‌پردازد) و هوش مصنوعی به کار گرفته شده است. همان طور که گفته شد در این منطق مقادیر صحیح پیوسته بین محدوده صفر تا ۱ قابل تغییر هستند. برای روشن‌شدن مطلب می‌توان مثال دما را مطرح نمود که هر تابع مقدار دما را در محدوده صفر تا ۱ مقداردهی می‌نماید.

در شکل ۲۱ مفاهیم سرد، گرم و داغ به‌وسیله توابعی منطبق بر مقیاس دما نشان داده شده است. در نقطه نشان داده شده در شکل، سه مقدار صحیح برای هر سه تابع وجود دارد به این صورت که پیکانی که به نقطه صفر اشاره می‌کند می‌تواند به‌صورت "not hot" تفسیر شود، پیکان در نقطه ۰٫۲بیانگر "slightly warm" و پیکان در نقطه ۰٫۸ بیانگر "fairly cold" می‌باشد. به‌طور معمول منطق فازی از قاعده IF-THEN به‌صورت (۱۳) استفاده می‌نماید

$$\text{if } variable \text{ is } property \text{ then } action \qquad (۱۳)$$

به‌طور مثال یک تعدیل‌کننده دما که داری یک پنکه می‌باشد به‌صورت (۱۴) در منطق فازی بیان می‌شود

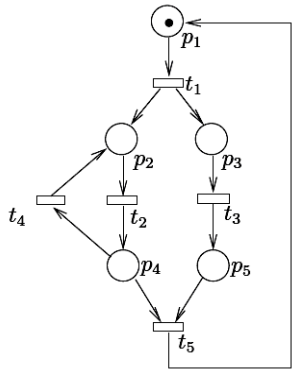
$$\begin{aligned} &\text{if } temprature \text{ is } very\ cold \text{ then } stopfan \\ &\text{if } temprature \text{ is } cold \text{ then } turn\ down\ \ fan \\ &\text{if } temprature \text{ is } normal \text{ then } maintain\ level \\ &\text{if } temprature \text{ is } hot \text{ then } speed\ up\ \ fan \end{aligned} \qquad (۱۴)$$

عملگرهای AND، OR و NOT نیز در منطق فازی با (۱۵) تا (۱۷) تعریف می‌شوند

$$\text{NOT}x = ۱ - \text{truth}(x) \qquad (۱۵)$$

$$x\text{AND}y = \text{minimum}(\text{truth}(x), \text{truth}(y)) \qquad (۱۶)$$

1. Multi-Valued Logic



شکل ۲۲: مثالی از شبکه پتری آماری.

$$x\text{OR}y = \text{maximum}(\text{truth}(x), \text{truth}(y)) \qquad (۱۷)$$

به‌صورت کلی می‌توان گفت سیستم‌های فازی دارای یک بخش ورودی، یک بخش پردازشی و یک بخش خروجی هستند. بخش ورودی معمولاً سنسورها و دکمه‌ها بوده که دارای یک تابع با مقادیر صحیح می‌باشند و بخش پردازشی که در آن یک قاعده بر اساس مقادیر صحیح از ورودی، فراخوانی و نتیجه‌ای را ایجاد می‌نماید. ترکیب نتایج ایجادشده بخش خروجی را می‌سازند. به‌طور مثال رابطه منطقی (۱۸) را برای یک سیستم ترموستات در نظر بگیرید:

$$\text{if } (temperature \text{ is } "cold") \text{ then } (heater \text{ is } "high") \qquad (۱۸)$$

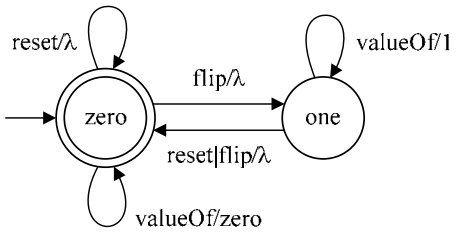
در (۱۸) "temperature" به‌عنوان ورودی بوده و "cold" یکی از مقادیر صحیح برای آن می‌باشد. این مجموعه خود باعث ایجاد تولید یک مجموعه فازی خواهد شد به این صورت که "heater" که در بخش خروجی است دارای مقدار "high" می‌گردد. با توجه به ویژگی‌های این گونه از سیستم‌ها نیاز به روش رسمی برای توصیف این گونه سیستم‌ها داریم که منطق فازی را پشتیبانی نمایند و به همین منظور برای این گونه نرم‌افزارها معیار نوع روش توصیف را لحاظ می‌نماییم.

٤-٧ نرم‌افزارهای احتمالی و آماری در مقابل نرم‌افزارهای قطعی

نرم‌افزارهای احتمالی و آماری[2] را در مقابل نرم‌افزارهای قطعی[3] در دسته جداگانه‌ای لحاظ می‌کنیم. این گونه نرم‌افزارها دارای ویژگی عدم قطعیت هستند که در ادامه به شرح آن می‌پردازیم.

- ویژگی عدم قطعیت: در این گونه نرم‌افزارها، احتمال انجام یک پروسه و کنش بر اساس یک عدد بیان می‌شود. به‌طور مثال احتمال تمام‌شدن پروژه در زمان تعیین‌شده ۹۵٪ است و در نتیجه احتمال پایان‌نیافتن آن در زمان تعیین‌شده ۵٪ (۰/۹۵–۱) می‌باشد. برای توصیف این گونه از نرم‌افزارها باید روش رسمی را انتخاب نمود که دارای امکان توصیف عدم قطعیت و احتمالات در مراحل توصیف یک نرم‌افزار باشد. از جمله روش‌های رسمی که برای نرم‌افزارهای دارای مشخصه احتمال و آمار مناسب می‌باشد می‌توان به شبکه‌های پتری آماری [۲۱] اشاره نمود. برای روشن‌شدن مطلب نمونه‌ای از شبکه پتری آماری در شکل ۲۲ آمده است. در این مثال نرخ وقوع هر گذار t_i، λ_i بوده و بر اساس قواعد شبکه‌های پتری آماری زمان وقوع گذارها به‌صورت نمایی توزیع شده است (رابطه (۱۹))

2. Stochastic Software
3. Deterministic Software



شکل ۲۳: توصیف فلیپ فلاپ با استفاده از ماشین حالت متناهی.

$$F_{Xi}(x) = ۱ - e^{-\lambda_i x} \qquad (۱۹)$$

گذار $t_۱$ در $M_۰ = (۱,۰,۰,۰)$ توانا بوده و مدت زمانی که طول می‌کشد تا این گذار آتش شود برابر با نرخ $\lambda_۱$ است. پس از آتش‌شدن این گذار و رسیدن به حالت $M_۱ = (۰,۱,۰,۰)$ دو گذار $t_۲$ و $t_۳$ توانا هستند اما با توجه به نرخ گذار آنها باید محاسبه نمود که کدام یک زودتر آتش می‌شوند. ابتدا فرض می‌کنیم که زمان وقوع $t_۲$ به‌صورت $(\lambda_۲ e^{-\lambda_۲ x})$ و زودتر از $t_۳$ با زمان وقوع $(\lambda_۳ e^{-\lambda_۳ x})$ آتش می‌گیرد. حال جهت محاسبه احتمال شرطی تک‌متغیره نیاز به محاسبه انتگرال ضرب این زمان‌ها در هم می‌باشد؛ پس جهت محاسبه می‌توانیم به‌صورت (۲۰) و (۲۱) عمل نماییم. با توجه به نتیجه (۲۱) احتمال وقوع گذار $t_۲$ قبل از $t_۳$ برابر با نرخ وقوع آن $(\lambda_۲)$ تقسیم بر جمع نرخ وقوع هر دو گذار $(\lambda_۲ + \lambda_۳)$ بوده و به همین ترتیب احتمال وقوع گذار $t_۳$ قبل از $t_۲$ برابر با نرخ وقوع آن $(\lambda_۳)$ تقسیم بر جمع نرخ وقوع هر دو گذار $(\lambda_۲ + \lambda_۳)$ است (رابطه (۲۱))

$$\begin{aligned} P[t_۲ \ fires \ first \ at \ M_۱] &= P[x_۲ < x_۳] \\ &= \int_۰^\infty (\int_۰^x \lambda_۲ e^{-\lambda_۲ y} \mathrm{d}y) \lambda_۳ e^{-\lambda_۳ x} \mathrm{d}x \\ &= \int_۰^\infty (۱ - e^{-\lambda_۲ x}) \lambda_۳ e^{-\lambda_۳ x} \mathrm{d}x = \frac{\lambda_۲}{\lambda_۲ + \lambda_۳} \end{aligned} \qquad (۲۰)$$

$$P[t_۳ \ fires \ first \ at \ M_۱] = \frac{\lambda_۳}{\lambda_۲ + \lambda_۳} \qquad (۲۱)$$

۸-٤ نرم‌افزارهای تبدیلی

نرم‌افزارهای تبدیلی[1] در هیچ یک از دسته‌های بالا قرار نمی‌گیرند. این نرم‌افزارها ورودی را از کاربر گرفته و خروجی را بر اساس الگوریتمی که بر روی داده‌های ورودی اعمال می‌کند، تولید نموده و کار خاتمه می‌یابد. در این گونه نرم‌افزارها بر اساس الگوریتم مد نظر روند کار دنبال می‌گردد و روش رسمی انتخابی برای این دسته می‌بایست دارای توانایی نمایش روند الگوریتمی برای نرم‌افزار تبدیلی باشد. برنامه‌های کاربردی مانند انواع نرم‌افزارهای گرافیکی، پردازشگر لغات و نرم‌افزارهای آموزشی از جمله نرم‌افزارهای تبدیلی هستند.

۵- روش‌های رسمی

به‌طور کلی روش‌های رسمی بر اساس چگونگی نحوه توصیف یک نرم‌افزار به چهار گروه اصلی تقسیم می‌شوند [۶]:

الف) مبتنی بر حالت[2]: یک حالت بیانگر وضعیت سیستم در نقطه یا بازه‌ای از زمان است. در این دسته از روش‌ها تمرکز بر روی حالت‌های سیستم بوده و توصیف اعمال سیستم (توابع) با تمرکز بر این که پس از اجرای این اعمال، چه تغییر حالت‌هایی در سیستم رخ می‌دهد، صورت می‌گیرد. در شکل ۲۳ توصیف یک فلیپ فلاپ که دارای دو حالت صفر و یک می‌باشد با استفاده از ماشین حالت متناهی[3] ارائه شده است.

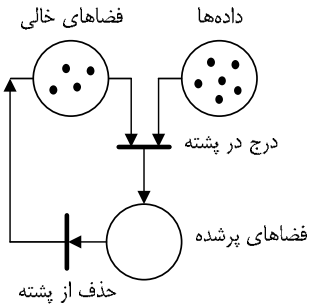
توصیف رسمی شکل ۲۳ با (۲۲) بیان شده است

$$\begin{aligned} &(\Sigma, S, S_۰, E, F) \ , \\ &\Sigma = \{\text{reset}, \text{flip}, \text{valueOf}\} \\ &S = \{\text{zero}, \text{one}\} \\ &S_۰ = \text{zero} \\ &E = \{(\text{zero}, \text{reset}) \to \text{zero}, \\ &\quad (\text{zero}, \text{flip}) \to \text{one}, \\ &\quad (\text{zero}, \text{valueOf}) \to \text{zero}, \\ &\quad (\text{one}, \text{reset}) \to \text{zero}, \\ &\quad (\text{one}, \text{valueOf}) \to \text{one}\} \\ &F = \text{zero} \end{aligned} \qquad (۲۲)$$

در گزاره اول (۲۲) ماشین حالت متناهی را به‌صورت یک چندتایی $(\Sigma, S, S_۰, E, F)$ معرفی نموده که در آن Σ الفبای ورودی، S مجموعه‌ای از حالت‌های ماشین، $S_۰$ حالت آغازین، E مجموعه‌ای از لبه‌ها (E زیرمجموعه‌ای از $S \Sigma S$ است) و F مجموعه‌ای از حالت‌هایی است که توسط ماشین پذیرفته می‌شوند.

ب) مبتنی بر رخداد[4]: یک رخداد پیامی است که بین محیط و سیستم مبادله می‌شود و در این دسته از روش‌ها تمرکز بر توصیف وقوع رخدادها و ترتیب وقوع آنها می‌باشد. توصیف رسمی رخداد به‌صورت (۲۳) در شبکه پتری است. در این رابطه M_i بیانگر حالت iام و نشانه‌های موجود در مکان‌ها در آن حالت بوده که با وقوع گذار t سیستم وارد حالت M_j می‌گردد

$$M_i \xrightarrow[t]{} M_j \qquad (۲۳)$$



شکل ۲۴: توصیف رخدادها در یک پشته با استفاده از شبکه‌های پتری.

به‌طور مثال برای توصیف یک پشته با استفاده از شبکه‌های پتری تمرکز ما بر بیان وقوع رخدادهایی مانند درج‌کردن و حذف‌کردن از پشته می‌باشد که ترتیب وقوع رخدادهای ذکرشده با استفاده از نشانه‌ها و گذارها در این روش مشخص می‌گردد (شکل ۲۴). توصیف رسمی رخداد push برای شکل ۲۴ در (۲۴) نشان داده شده است

1. Transformation Software
2. State - Based
3. Finite State Machine
4. Event - Based

$$BirthdayBook$$
$$known : \mathbb{P}\, NAME$$
$$birthday : NAME \pfun DATE$$
$$known = \mathrm{dom}\, birthday$$

$$AddBirthday$$
$$\Delta BirthdayBook$$
$$name? : NAME$$
$$date? : DATE$$
$$name? \notin known$$
$$birthday' = birthday \cup \{name? \mapsto date?\}$$

شکل ۲۵: تعریف پیمانه‌ای کتاب تاریخ تولد در Z.

$$M_{\cdot} = (۴,۶,۰) \underset{push}{\rightarrow} M_{۱} = (۳,۵,۱) \qquad (۲۴)$$

ج) منطقی: این دسته از روش‌ها از قوانین و نمادهای منطقی (منطق رتبه اول[1] یا بالاتر) جهت توصیف و مدل‌سازی سیستم بهره می‌برند. منطق گزاره‌ای[2] یک شاخه از منطق بوده که در آن از گزاره‌ها و از علایم ربط منطقی برای ارتباط برقرار کردن بین گزاره‌ها استفاده می‌شود [۲۲]. در این منطق هر گزاره می‌تواند دارای مقدار صحیح یا غلط باشد. به‌طور مثال اگر هوا ابری باشد آنگاه باران می‌بارد. حساب شرطی یا منطق شرطی، بسط‌یافته منطق گزاره‌ای بوده که در آن نمادهای گزاره، زیرموضوع[3] و کمیت‌سنج‌ها از هم جدا شده‌اند. به‌طور مثال در منطق گزاره‌ای می‌توان نماد P را به گزاره "All men are mortal" اختصاص داد اما در منطق شرطی می‌توان گزاره $M(x)$ را تعریف نمود که جایگزین مردن برای زیرموضوع x خواهد بود. x را می‌توان به کمیت‌سنج جهانی مقید نمود (For all) و جمله منطقی را در نهایت به‌صورت $\text{All}\, x.M(x)$ بیان کرد. تعداد زیرموضوع‌ها در منطق شرطی رتبه اول یکی بوده و در واقع تک‌متغیره می‌باشد اما در منطق‌های شرطی رتبه بالاتر[4] تعداد زیرموضوع‌ها و متغیرها می‌تواند بیشتر از یکی باشند.

د) جبری: در این گونه از روش‌های رسمی، سیستم به‌وسیله قوانین جبری یعنی مجموعه‌ها و عملیات روی مجموعه‌ها مدل شده و رفتار رخدادها (توابع) با استفاده از معادلات توصیف می‌گردند. هر توصیف جبری شامل دو بخش است: (۱) قسمتی برای توصیف داده (با استفاده از مجموعه‌ها) و عملیات بر روی آن و (۲) بخش توصیف عملیات با استفاده از بدیهیات. به‌طور مثال برای توصیف اعداد طبیعی N با معرفی صفر و با دو عمل تعیین عضو بعدی و جمع به‌صورت (۲۵) و (۲۶) نشان داده می‌شوند؛ روابط (۲۵) و (۲۶) به‌ترتیب نشان‌دهنده عملگرها (توابع) و اصول هستند

$$\begin{aligned}&\text{sorts Nat}\\&\text{opns}\\&\quad \cdot :\rightarrow \text{Nat}\\&\quad \text{succ} : \text{Nat} \rightarrow \text{Nat}\\&\quad + : \text{Nat}, \text{Nat} \rightarrow \text{Nat}\end{aligned} \qquad (۲۵)$$

$$\begin{aligned}&\text{sorts Nat}\\&\text{axioms}\\&\quad \text{for all } x : \text{Nat}\\&\qquad x + \cdot = x\\&\quad \text{for all } x : \text{Nat}, y : \text{Nat}\\&\qquad \text{succ}(x) + y = \text{succ}(x + y)\end{aligned} \qquad (۲۶)$$

در این بخش از گروه‌های مختلف نام برده شده، نمونه‌ای انتخاب و سپس به ارزیابی آن روش رسمی بر اساس معیارهای ارائه‌شده در بخش سوم پرداخته شده است. از گروه اول به روش Z، از گروه دوم به Event-B، از گروه سوم به منطق زمانی بی‌درنگ و از گروه چهارم به CSP می‌پردازیم.

۵-۱ روش مبتنی بر حالت، زبان Z

ایده Z برای اولین بار در سال ۱۹۷۷ به‌وسیله Abrial که عضو گروه تحقیقاتی برنامه‌نویسی دانشگاه آکسفورد بود، ارائه گردید [۲۳]. این زبان یک زبان رسمی توصیفی بر پایه تئوری مجموعه‌ها و منطق شرطی رتبه اول می‌باشد.

نوع داده: این روش موجودیت‌های سیستم را با نوع داده تعریف می‌نماید که انواع داده پیش‌تعریف برای این روش شامل نوع داده صحیح و نوع داده طبیعی می‌باشد. به‌طور مثال برای توصیف جهت حرکت آسانسور (دو حالت بالا و پایین) و برای دکمه آسانسور (دو حالت خاموش و روشن) دو نوع داده تعریف می‌کنیم (روابط (۲۷) و (۲۸)). جهت حرکت آسانسور (رابطه (۲۸)) و دکمه آن (رابطه (۲۷)) از جنس داده‌های شمارشی است. رابطه (۲۹) که قید روی مقادیر داده‌ها را نشان می‌دهد، اظهار می‌دارد که شماره طبقات باید از نوع عدد صحیح و مقدار بزرگ‌تر از صفر باشد

$$\text{SWITCH} ::= \text{on} \mid \text{off} \qquad (۲۷)$$

$$\text{MOVE} ::= \text{up} \mid \text{down} \qquad (۲۸)$$

$$\mid \frac{\text{FLOORS} : \mathbb{N}}{\text{FLOORS} > \cdot} \qquad (۲۹)$$

فضای حالت: مجموعه متغیرهای دارای نوع که در توصیف سیستم استفاده می‌شوند، فضای حالت سیستم را مشخص می‌سازند.

پیمانه: تعریف حالت‌ها و چگونگی تغییر آنها به‌وسیله عملیات، توسط ساختارهایی به نام شما[5] (شکل ۲۵) توصیف می‌شوند [۲۴]. روش Z دارای امکان توصیف پیمانه‌ها با استفاده از شما است. به‌طور مثال در شکل ۲۵ در بخش ابتدایی توصیف یک کتاب جهت ثبت، تاریخ تولد و اسامی افراد معرفی شده و در قسمت بعدی با استفاده از نماد Δ به این بخش ارجاع شده است.

وراثت: روش Z دارای نماد نحوی جهت پشتیبانی از مفاهیم وراثت و شیءگرایی نبوده اما Object - Z [۲۵] که نسخه بسط‌یافته زبان Z است، از مفاهیم شیءگرایی پشتیبانی می‌نماید.

زمان: در روش Z فقط به تقدم و تأخر رخدادها (توصیف حالت فعلی و بیان یک کنش جهت رفتن حالت بعدی) توجه شده و فاصله زمانی رخدادها مورد توجه قرار نمی‌گیرد.

اثبات‌پذیری: با توجه به استفاده از نمادهای ریاضی در توصیف‌های

1. First-Order Logic
2. Propositional Logic
3. Subject
4. Higher-Order Predicate Logic
5. Schema

این روش امکان اثبات توصیف‌ها برای این روش وجود دارد. به‌طور مثال شکل ۲۵ با مجموعه (۳۰) اثبات می‌شود

$$\begin{aligned} known' &= \mathrm{dom} birthday' \\ &= \mathrm{dom}(birthday \cup \{name? \to date?\}) \\ &= \mathrm{dom} birthday \cup \mathrm{dom}\{name?\} \\ &= known \cup \{name?\} \end{aligned} \quad (۳۰)$$

پشتیبانی ابزاری: ابزارهای مختلفی برای روش Z طراحی شده که بسیاری از آنها در حد پروژه‌های دانشجویی بوده و تعداد اندکی از آنها دارای کیفیتی در حد یک محصول و ارائه‌دهنده یک محیط مجتمع مناسب جهت استفاده توسعه‌دهندگان سیستم‌ها می‌باشد. پروژه CZT[1] [۲۶] که توسط Andrew Martin در آزمایشگاه کامپیوتر دانشگاه آکسفورد پایه‌گذاری شده شامل مجموعه ابزار جهت ویرایش، بررسی نوع و پویانمایی[2] برای توصیف‌های رسمی نوشته‌شده به زبان Z است که برخی از نسخ بسط‌یافته Z مانند object - Z، Circus و TCOZ را نیز پشتیبانی می‌نماید. از این ابزار می‌توان جهت توصیف دقیق نیازمندی‌های سیستم یا توصیف رفتاری سیستم و تحلیل رفتار از طریق اثبات، پویانمایی، تولید آزمون و وارسی استفاده نمود.

۵-۲ روش مبتنی بر رخداد، Event - B

روش Event - B نیز برای اولین بار توسط Abrial [۲۷] در سال ۱۹۹۶ ارائه گردیده است. در مقایسه با زبان Z این روش بیشتر بر پالایش کد تمرکز داشته و به همین خاطر تبدیل توصیف‌های آن به پیاده‌سازی راحت‌تر صورت می‌گیرد و همچنین در مقایسه با زبان Z، این روش دارای ابزار توسعه‌یافته‌تری است.

مدل: روش Event - B از یک مفهوم پایه به نام مدل برای توصیف رسمی خود استفاده می‌نماید. یک مدل شامل عناصر متعدد از نوع ماشین[3] و متن[4] است. ماشین‌ها معرف قسمت پویای مدل، وضعیت کنونی سیستم و دربرگیرنده متغیرها، قیود، قضایا و رخدادها است. مفاهیم، معرف قسمت ایستای مدل و دربرگیرنده داده‌ها، ثابت‌ها و وجوه بدیهی است که در طول توسعه سیستم استفاده می‌شود.

رخداد: روش Event - B (همان‌طور که نام آن نشان می‌دهد) با استفاده از رخدادها به توصیف یک سیستم می‌پردازد. از این رو در دسته روش‌های مبتنی بر رخداد قرار می‌گیرد. در این روش رسمی، رخدادها بیانگر عملیات وابسته به ماشین‌ها بوده و از طریق همین رخدادها سیستم با محیط پیرامون خود تعامل خواهد داشت. یک رخداد می‌تواند شامل مجموعه‌ای از پیش‌شرط‌ها و مجموعه‌ای از کنش‌ها بوده که به‌وسیله آنها متغیرها مقداردهی می‌شوند. یک رخداد در این روش با (۳۱) بیان می‌شود. در این رابطه S بیانگر کنش‌ها و v نماینده متغیرهایی است که در طول رخداد مقداردهی می‌شوند

$$evt \equiv \mathrm{BEGIN}(v)\mathrm{END} \quad (۳۱)$$

پیمانه: روش Event - B با استفاده از مدل‌ها، توانایی توصیف بخش‌های مختلف سیستم را به‌صورت پیمانه‌ای داراست. به‌طور مثال در (۳۲) و (۳۳) فاکتور ۱ توصیف می‌شود. این فاکتور شامل نوعی محصول (Product۱) است و دارای مجموعه سفارش‌ها و موقعیت آنها (سفارش‌های در حال انتظار و سفارش‌های فاکتورشده) است و include نشان می‌دهد که این فاکتور شامل Product۱ است. Product۱ خود جزء دسته محصول‌ها بوده و در توصیف آن از متغیرهای محصول و تعداد موجودی انبار با این قید که هر دو در ابتدا دارای مقدار صفر هستند؛ استفاده شده است

MACHINE
*Invoicing*۱
SETS
ORDER;
STATUS = {Order pending, order invoiced}
RESPONSE = {Updated, Not updated}
INCLUDES
*Product*۱ (۳۲)

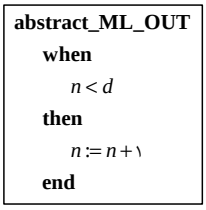
MACHINE
*Product*۱
SETS
PRODUCT
VARIABLES
product, *quantity in stock* (۳۳)
INVARIANT
$product \subseteq PRODUCT \wedge$
$quantity\ in\ stock \in product \to \mathbf{NAT}$
INITIALISATION
$product := \varnothing \| quantity\ in\ stock := \varnothing$

پالایش: ویژگی کلیدی این روش استفاده از پالایش مدل بوده که توسعه افزایشی مدل‌ها را از مدل متنی اولیه فراهم می‌سازد. در این روش در هر توسعه، صحت توصیف‌ها در مدل پالایش‌یافته جدید با استفاده از قواعد استنتاجی تضمین می‌شود. برای مثال یک پل عبور یک‌طرفه از جزیره اصلی به یک جزیره فرعی را در نظر می‌گیریم. این پل حداکثر ظرفیت عبور هم‌زمان d ماشین را دارد و n تعداد واقعی ماشین‌ها در یک لحظه از زمان در روی پل و جزیره فرعی می‌باشد و متغیر a تعداد ماشین‌هایی است که از روی پل به سمت جزیره فرعی حرکت کرده و متغیر b تعداد ماشین‌هایی است که در جزیره فرعی هستند و متغیر c تعداد ماشین‌هایی است که از روی پل به سمت جزیره اصلی در حرکت هستند. توصیف اولیه با توجه به موارد گفته‌شده به‌صورت (۳۴) تا (۳۶) است.

قید ۱، ۲ و ۳ در (۳۵) به حقیقی‌بودن جنس سه متغیر a، b و c اشاره دارد. قید ۴ در (۳۶) اظهار می‌دارد که جمع تعداد ماشین‌ها بر روی پل و جزیره برابر با n است و چون پل یک‌طرفه است، به‌طور هم‌زمان یا تعداد ماشین‌ها روی پل به سمت جزیره (a) برابر صفر بوده یا تعداد ماشین‌ها روی پل به سمت بیرون جزیره (c) برابر صفر است (قید ۵). رخداد خروج ماشین‌ها از جزیره اصلی (ML_OUT) را در شکل ۲۶ توصیف می‌کنیم

constant : d
variables : a, b, c (۳۴)

1. The Community Z Tools
2. Animating
3. Machine
4. Context



شکل ۲۶: توصیف یک رخداد در روش Event-B.

$$\begin{aligned} &\mathbf{INV\_۱}: a \in N \\ &\mathbf{INV\_۲}: b \in N \\ &\mathbf{INV\_۳}: c \in N \end{aligned} \quad (۳۵)$$

$$\begin{aligned} &\mathbf{INV\_۴}: a+b+c = n \\ &\mathbf{INV\_۵}: a = ۰ \vee c = ۰ \end{aligned} \quad (۳۶)$$

شکل ۲۶ اظهار می‌دارد که رخداد خروج ماشین از جزیره اصلی و حرکت به سمت جزیره فرعی به معنای افزایش یک واحدی تعداد مجموع ماشین‌ها (n) است به این شرط که تعداد آنها از حداکثر ظرفیت (d) تجاوز نکند. پالایش شکل ۲۶ در شکل ۲۷ نشان داده شده است.

شکل ۲۷ اظهار می‌دارد که رخداد، خروج ماشین از جزیره اصلی و حرکت به سمت جزیره فرعی زمانی صورت می‌گیرد که تعداد ماشین‌های روی پل که به سمت جزیره فرعی در حرکت هستند (a)، به همراه ماشین‌های روی پل (b) کمتر از حداکثر ظرفیت (d) باشد. چون پل یک‌طرفه است، اگر ماشینی قصد حرکت به سوی جزیره فرعی را دارد باید تعداد ماشین‌های در حال خروج از جزیره فرعی (c) صفر باشد و با حرکت این ماشین، به تعداد ماشین‌هایی که به سمت جزیره فرعی حرکت می‌کنند (a) یکی اضافه شود.

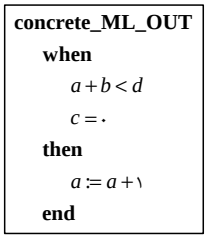
همروندی و همگامی: یکی از نقاط ضعف این روش عدم پشتیبانی توصیف فرایندهای همگام و همروند است.

پشتیبانی ابزاری: برای این روش ابزاری با نام Rodin Platform [۲۸] ارائه شده است که این ابزار، تجمیعی از چندین ابزار جهت پوشش مراحل مختلف توسعه نرم‌افزار می‌باشد. این ابزار مراحل توصیف، تحلیل، اثبات، بررسی مدل و مستندسازی را به‌صورت خودکار پشتیبانی می‌نماید.

۳-۵ روش منطقی: منطق زمانی بی‌درنگ

منطق زمانی بی‌درنگ یک زبان منطق رتبه اول بوده که اولین بار توسط Jahanian و Mok [۲۹] در سال ۱۹۸۶ ارائه گردیده است. این روش در ابتدا برای استدلال درباره ویژگی‌های زمانی سیستم‌های بلادرنگ ایجاد شده است. در این روش با یک گزاره، رخدادهای یک سیستم به زمان وقوع آنها وابسته شده و یک سیستم بلادرنگ به‌صورت کلاسی از انواع رخدادها مدل می‌گردد. مشکل اصلی این روش این است که وارسی فرمول‌های منطق زمانی بی‌درنگ کار بسیار مشکلی بوده چون تعداد نامحدودی از وقوع رخدادها بایستی در نظر گرفته شوند [۲۹].

منطق و رخداد: در روش منطق زمانی بی‌درنگ، رخدادهای یک سیستم به‌وسیله فرمول‌های منطقی توصیف می‌شوند و به همین خاطر این روش هم در دسته روش‌های مبتنی بر منطق و هم در دسته روش‌های مبتنی بر رخداد قرار می‌گیرد. به‌طور مثال می‌خواهیم یک سیستم که شامل تولیدکننده و مصرف‌کننده است را با این روش توصیف نماییم. در این مثال تولیدکننده در ابتدای هر



شکل ۲۷: پالایش یک رخداد در روش Event-B.

فاصله زمانی به اندازه p واحد، پیامی را در حافظه قرار می‌دهد. مصرف‌کننده، پیام را از حافظه پاک نموده و در هر فاصله زمانی از p پیام برداشته‌شده را مورد پردازش قرار می‌دهد. علاوه بر این مصرف‌کننده برای پردازش پیام به c واحد زمانی نیاز دارد. از آنجایی که تولیدکننده و مصرف‌کننده روی پردازنده‌های متفاوتی قرار گرفته‌اند، ساعت مصرف‌کننده به اندازه α واحد زمانی عقب‌تر از ساعت تولیدکننده می‌باشد. توصیف این سیستم با روش منطق زمانی بی‌درنگ با (۳۷) تا (۳۹) ارائه می‌شود

$$Producer: \forall i: @(P,i) = i \times p \quad (۳۷)$$

$$\begin{aligned} Consumer: \forall i: & \; i \times p + \alpha \le @(\uparrow Q, i) \wedge @(\downarrow Q, i) \\ & \le (i+۱) \times p + \alpha \\ & \forall i: @(\uparrow Q, i) + c \le @(\downarrow Q, i) \end{aligned} \quad (۳۸)$$

$$Clock\ Skew: \alpha < p \quad (۳۹)$$

رخداد P در توصیف این سیستم نمایانگر درج یک پیام در حافظه توسط تولیدکننده در فواصل زمانی p بوده و رخداد $\uparrow Q$ نشان‌دهنده برداشته‌شدن پیام و شروع پردازش یک پیام توسط مصرف‌کننده و رخداد $\downarrow Q$ بیانگر کامل‌شدن پردازش پیام توسط مصرف‌کننده می‌باشد.

زمان: در این روش همان‌طور که در مثال تولیدکننده – مصرف‌کننده دیده شد امکان توصیف زمان به‌صورت کمی وجود دارد. برای مثال در توصیف ارائه‌شده (خط چهارم) بیان شده که مدت زمان شروع مصرف تا پایان مصرف، c واحد زمانی می‌باشد.

قابلیت اثبات: از مزایای دیگر این روش بهره‌بردن از فرمول‌های منطقی بوده که قابلیت اثبات را برای توصیف‌های این روش فراهم می‌سازد. به طور مثال برای یک سیستم تولیدکننده – مصرف‌کننده که در (۳۷) تا (۳۹) توصیف گردید، قید ایمنی برای گزاره "عملیات شروع مصرف $(\uparrow Q)$ نمونه iام توسط مصرف‌کننده باید قبل از تولید نمونه $(i+۱)$ام توسط تولیدکننده صورت گیرد" را با (۴۰) نشان می‌دهیم

$$\forall i: @(\uparrow Q, i) \le @(P, i+۱) \quad (۴۰)$$

حال با استفاده از برهان خلف به اثبات (۴۰) می‌پردازیم. توصیف‌های تولیدکننده – مصرف‌کننده را با (SP) $(a-۴۱)$ تا $(e-۴۱)$ و نقیض قید ایمنی را با (SA) $(f-۴۱)$ نشان می‌دهیم

$$\begin{aligned} &SP: @(P,i) = i \times p && (a) \\ &i \times p + \alpha \le @(\uparrow Q, i) && (b) \\ &@(\downarrow Q, i) \le (i+۱) \times p + \alpha && (c) \\ &@(\uparrow Q, i) + c \le @(\downarrow Q, i) && (d) \\ &\alpha < p && (e) \\ &SA: @(P, i+۱) < @(\uparrow Q, i) && (f) \end{aligned} \quad (۴۱)$$

نماد	معنی	مثال
$=$	برابر است	$x = x$
$\neq$	متفاوت است از	$x \neq x + 1$
$\square$	اثبات	
$P \wedge Q$	و منطقی	$x \leq x + 1 \wedge x \neq x + 1$
$P \vee Q$	یا منطقی	$x \leq y \vee y \leq x$
$\neg P$	نه منطقی	$\neg 3 \geq 5$
$P \Rightarrow Q$	if p then q p اگر فقط اگر q	$x < y \Rightarrow x \leq y$
$P \equiv Q$		$x < y \equiv y > x$
$\exists x \bullet P$	وجود دارد x ای مانند p	$\exists x \bullet x > y$
$\forall x \bullet P$	برای همه xها p صادق است	$\forall x \bullet x < x + 1$
$\exists x : A \bullet P$	وجود دارد xای در مجموعه A مانند p	
$\forall x : A \bullet P$	برای همه xها در مجموعه A p صادق است	

شکل ۲۸: نمادهای روش جبری CSP.

$$\begin{array}{ll} a.\ (i+1)\times p < @(\uparrow Q,i) & (a-41),(f-41) \\ b.\ @(\downarrow Q,i) < @(\uparrow Q,i)+\alpha & (a-42),(c-41) \\ c.\ @(\uparrow Q,i)+c < @(\uparrow Q,i)+\alpha & (b-42),(d-41) \\ d.\ c < \alpha & (c-42) \end{array} \quad (42)$$

نامساوی موجود در گام d در (۴۲) نقیض a تا f در (۴۱) بوده و از این رو اگر $\alpha \leq c$ باشد، $SP \rightarrow SA$ صحیح است.

پیمانه و وراثت: روش منطق زمانی بی‌درنگ دارای امکان شکستن توصیف‌های بزرگ به بخش‌های کوچک‌تر نبوده و به همین دلیل توانایی پیمانه‌سازی ندارد. علاوه بر آن، نماد نحوی جهت پشتیبانی از مفاهیم شیءگرایی برای این روش ارائه نشده است.

پشتیبانی ابزاری: این روش دارای ابزار جهت توصیف یا تغییر توصیف و اعتبارسنجی نبوده و تنها ابزار موجود برای این روش Verify۴ است [۳۰] که در قسمت وارسی توصیف به کار می‌رود.

۵-۴ روش جبری: CSP

روش CSP [۳۱] در سال ۱۹۷۸ توسط C. A. Hoare معرفی گردید. این روش یک روش رسمی با نمادهای متنی جهت توصیف و تحلیل سیستم‌های همروند بوده که در آن مؤلفه‌های فرایندها از طریق ارتباط بر یکدیگر اثر متقابل دارند [۳۲]. این روش متعلق به خانواده نظریه ریاضی همروند بوده که به‌عنوان پردازش جبری یا Process Calculus شناخته شده است. این روش می‌تواند همگامی بین فرایندهای همروند (همگامی زمانی اتفاق می‌افتد که دو فرایند قصد داشته باشند به‌صورت همگام پیامی را ارسال و دیگری دریافت نماید) را مدل نماید [۳۲]. چون این روش بر پایه نظریه ریاضی است، می‌تواند جهت استدلال نیز مورد استفاده قرار گیرد.

جبر مبتنی بر رخداد: نوع نمادهای مورد استفاده در این روش هم متنی و هم ریاضی می‌باشد. تمرکز این روش برای توصیف یک نرم‌افزار با استفاده از قوانین جبری برای بیان فرایندها بوده و آنچه در توصیف یک نرم‌افزار مورد اهمیت قرار می‌گیرد رخدادهای موجود می‌باشند. به این خاطر این روش هم در دسته روش‌های مبتنی بر رخداد و هم در دسته روش‌های جبری قرار می‌گیرد. نمادهای منطقی که در این روش استفاده می‌شوند در شکل ۲۸ ارائه شده است.

پیمانه‌سازی: در روش CSP امکان شکستن توصیف‌های بزرگ به پیمانه‌های کوچک‌تر و امکان توصیف سلسله مراتبی وجود دارد [۳۲]. به‌طور مثال می‌توان ابتدا بیان کلی از یک نرم‌افزار ارائه نمود و سپس در توصیف‌های بعدی جزئیات را مشخص کرد. مثالی برای روشن‌شدن مطلب ارائه می‌شود. اولین گزاره (۴۳) اظهار می‌دارد که سیستم عامل (*OP*) از یک نرم‌افزار دسته‌ای (*BATCH*) و یک سامانه ورودی- خروجی (*IOSYSTEM*) تشکیل شده است و دومین گزاره (۴۳) سامانه ورودی- خروجی را توصیف می‌کند. در این سامانه ورودی- خروجی یک فایل سیستمی (*filesys*) بین خط پرینتر (*lp*) از قسمت خروجی و بین کارت‌خوان (*cr*) از قسمت ورودی به اشتراک گذاشته شده است

$$\begin{aligned} &OP = IOSYSTEM \;//\; BATCH\,۴ \\ &IOSYSTEM = SH : (filesys : FILESYS)\;// \\ &\quad (lp : OUTSPOOL \,\|\, cr : INSPOOL) \end{aligned} \quad (43)$$

زمان: در این روش امکان بیان فاصله زمانی بین رخدادها وجود نداشته و فقط به تقدم و تأخر رخدادها توجه شده است.

همروندی و همگامی: در روش CSP امکان نمایش همروندی و همگامی بین فرایندها وجود دارد [۳۱]. به‌طور مثال عبارت $P[\![\{a\}]\!]Q$ بیانگر این است که P و Q هر دو باید قادر به اجرای رخداد a قبل از این که اتفاق بیافتد، باشند.

پشتیبانی ابزاری: دو ابزار FDR[1] و PAT[2] برای این روش ارائه شده‌اند [۳۳] که ابزار FDR جهت بررسی مدل کاربرد داشته و ابزار PAT نیز غالباً جهت اعتبارسنجی توصیف‌ها به کار می‌رود.

۵-۵ تعیین برازندگی و تناسب روش‌های رسمی

پس از بررسی کلی روش‌های رسمی و ارائه یک نمونه از هر یک، به تعیین برازندگی[3] این روش‌ها می‌پردازیم. برازندگی روش‌های رسمی که نشان‌دهنده توانایی یک روش رسمی با توجه به معیارهای ارائه‌شده در بخش سوم است، به‌صورت جدول ارائه می‌شود (جداول ۱ و ۲). در این جداول هر روش رسمی نماینده گروهی از روش‌های رسمی است که در ستون دوم جدول (جدول ۱) نام گروه بیان شده است.

پس از تعیین برازندگی بر اساس طبقه‌بندی نرم‌افزارها در بخش چهارم، تناسب هر معیار با نرم‌افزار مورد توصیف به‌صورت جدول ارائه می‌شود (جدول ۳). در نهایت بر اساس تعیین برازندگی و تناسب در جداول ۱ تا ۳، جدول راهبرد (جدول ۴) ارائه می‌شود که در آن به تعیین تناسب روش رسمی برای توصیف یک طبقه از نرم‌افزار پرداخته می‌شود.

در ادامه، به شرح راهبرد انتخاب و تصمیم می‌پردازیم. با گرفتن اشتراک بین جداول ۱ تا ۳، جدول ۴ را می‌سازیم. در این جدول تعیین می‌کنیم که از بین ۵ روش رسمی مورد بررسی، کدام یک برای چه نوع نرم‌افزاری مناسب هستند. لازم به ذکر است که هر یک از ۵ روش رسمی نماینده گروهی خاص از روش‌های رسمی است. به‌طور مثال برای توصیف یک نرم‌افزار توزیع‌شده می‌توان از روش CSP بهره برد و یا برای

1. Failures/Divergence Refinement
2. Process Analysis Toolkit
3. Fitness

جدول ۱: جدول برازندگی- بخش اول (✓: دارد، ×: ندارد).

معیار / نام روش	نوع روش توصیف	نمایش همروندی	نمایش همگامی	نمایش زمان	نمایش عدم قطعیت	نمایش ایمنی و حیات
Z	مبتنی بر حالت	×	×	×	×	✓
Event-B	مبتنی بر رخداد	×	×	✓	×	✓
منطق زمانی بی‌درنگ	منطقی- مبتنی بر رخداد	×	×	✓	×	✓
CSP	جبری	✓	✓	×	×	✓

جدول ۲: جدول برازندگی- بخش دوم.

معیار / نام روش	خوانایی (نوع توصیف)	خوانایی (ساختاری)	قابلیت اجرا	اثبات‌پذیری	بررسی مدل	امکان مستندسازی	قابلیت نگاشت	پشتیبانی ابزاری
Z	۲	۱	۲	۲	۲	۲	۰	۲
Event-B	۲	۱	۲	۲	۱	۲	۲	۲
منطق زمانی بی‌درنگ	۲	۰	۱	۱	۲	۰	۰	۱
CSP	۲	۱	۲	۲	۲	۰	۰	۰

جدول ۳: جدول تناسب (✓: لازم است، ×: لازم نیست).

معیار / نوع نرم‌افزار	نوع روش توصیف	نمایش همروندی	نمایش همگامی	نمایش زمان	نمایش عدم قطعیت	نمایش ایمنی و حیات
موازی	×	✓	×	×	×	×
بلادرنگ	×	×	×	✓	×	✓
توزیع‌شده	×	✓	✓	×	×	✓
هوش مصنوعی	✓	×	×	×	×	×
خبره	✓	×	×	×	×	×
منطقی	✓	×	×	×	×	×
دارای احتمال و آمار	×	×	×	×	✓	×
تبدیلی	×	×	×	×	×	×

جدول ۴: جدول راهبرد (✓: مناسب است، ×: مناسب نیست).

نرم‌افزار / نام روش	موازی	بلادرنگ	توزیع‌شده	هوش مصنوعی	خبره	منطقی	دارای احتمال و آمار	تبدیلی
Z	×	×	×	×	×	×	×	✓
Event-B	×	✓	×	×	×	×	×	✓
منطق زمانی بی‌درنگ	×	✓	×	✓	✓	✓	×	✓
CSP	✓	×	✓	×	×	×	×	✓

توصیف یک نرم‌افزار هوش مصنوعی می‌توان از روش منطق زمانی بی‌درنگ استفاده نمود. برای توضیح نحوه اشتراک‌گیری بین جداول ۱ تا ۳ به بررسی یک ردیف انتخابی از جداول ذکرشده می‌پردازیم. برای نمونه می‌خواهیم مناسب‌بودن روش Event-B را برای توصیف نرم‌افزارهای بلادرنگ مورد بررسی قرار دهیم. ابتدا ویژگی‌های مورد نیاز یک روش رسمی برای توصیف نرم‌افزارهای بلادرنگ را از جدول ۳ (جدول تناسب) استخراج می‌نماییم. بر اساس نتایج جدول ۳ دیده می‌شود که برای توصیف یک نرم‌افزار بلادرنگ نیاز به روش رسمی داریم که بتواند دارای نمایش زمان و نمایش ایمنی و حیات باشد. بر این اساس به جداول ۱ و ۲ (جداول برازندگی روش‌های رسمی) مراجعه نموده و پشتیبانی روش Event-B را از دو ویژگی ذکرشده مورد بررسی قرار می‌دهیم. همان‌طور که در جدول ۱ دیده می‌شود. روش Event-B دارای نمایش زمان و نمایش ایمنی و حیات می‌باشد و بر همین اساس در جدول ۴ (جدول راهبرد) در تقاطع ستون مربوط به نرم‌افزارهای بلادرنگ و روش Event-B علامت ✓ را برای بیان مناسب‌بودن این روش در توصیف نرم‌افزارهای بلادرنگ درج می‌کنیم.

۶- مطالعه موارد

حال با توجه به ارزیابی‌های صورت‌گرفته بر روی چهار روش رسمی انتخابی که از گروه‌های مختلف روش‌های رسمی انتخاب شده‌اند، برای روشن‌شدن و درک بهتر مفاهیم، در ذیل، دو مورد را مطالعه می‌کنیم که در این موارد تلاش شده برای نرم‌افزار مورد نظر، روش رسمی مناسبی از میان این چهار روش مورد بررسی جهت توصیف یا تحلیل انتخاب نمود.

مورد ۱: توصیف نرم‌افزار دستگاه تنظیم ضربان قلب. این دستگاه دارای الکترودهایی است که به ماهیچه‌های قلب وصل شده و با تولید پالس‌های الکتریکی ضربان قلب را تنظیم می‌نماید.

گام ۱: تعیین نوع نرم‌افزار. از آنجا که این نرم‌افزار مدام در تعامل با ضربان گرفته‌شده از بیمار است، بر اساس خصوصیات ذکرشده در بخش چهارم برای نرم‌افزارهای بلادرنگ این نرم‌افزار در دسته نرم‌افزارهای واکنشی و بلادرنگ قرار می‌گیرد (ویژگی اصلی این گونه نرم‌افزارها تعامل دائم با محیط و پاسخ‌گویی بر اساس تغییرات در محیط می‌باشد (شکل ۱۷). بر اساس ویژگی‌های ذکرشده برای این گونه نرم‌افزارها نیاز به روش رسمی خواهیم داشت که بتواند مفاهیم زمان و ایمنی و حیات را نمایش دهد.

گام ۲: تعیین روش رسمی مناسب. حال با توجه به موارد بالا از میان چهار روش رسمی مورد بررسی، به نتایج ارزیابی این روش‌ها مراجعه می‌کنیم و بر اساس نتایج به‌دست آمده در بخش چهارم تنها دو روش رسمی منطق زمانی بی‌درنگ و Event - B دارای نمایش زمان و ایمنی و حیات هستند. از این رو این دو روش برای توصیف این نرم‌افزار مناسب بوده و سایر روش‌ها از لیست انتخاب حذف می‌گردند.

مورد ۲: توصیف یک نرم‌افزار جستجو برای یافتن یک مورد مناسب در اجاره بهای یک خانه، این نرم‌افزار با پرسیدن سؤالات متعدد از کاربر باید مورد مناسب را از میان پایگاه اطلاعات پیدا و معرفی نماید. از جمله سؤالات می‌توان به متراژ خانه، میزان اجاره، تعداد اتاق و موارد دیگر اشاره کرد.

گام ۱: تعیین نوع نرم‌افزار. با بررسی مشخصات نرم‌افزار مشخص می‌شود این نرم‌افزار در تلاش برای یافتن پاسخ مناسب از پایگاه دانش خود می‌باشد اما پاسخ‌هایی که این نرم‌افزار ارائه می‌نماید با توجه به تناسب موارد مد نظر (متراژ، میزان اجاره و موارد دیگر) با فاکتور عددی اطمینان بیان خواهد شد و علاوه بر آن برای رسیدن به موارد مناسب لازم است تا زنجیره رو به عقب برای این موارد موجود در پایگاه دانش طی شده و بر اساس آن استدلال صورت گیرد. بنابراین این نرم‌افزار در دسته نرم‌افزارهای خبره قرار می‌گیرد.

گام ۲: تعیین روش رسمی مناسب. با توجه به نوع نرم‌افزار نیاز به روش رسمی است که نوع توصیف آن منطقی باشد. حال به ارزیابی چهار روش رسمی مورد نظر در بخش پنجم مراجعه می‌کنیم. از میان لیست مد نظر، تنها روش منطق زمانی بی‌درنگ دارای معیار مورد نظر بوده و سایر روش‌ها از فهرست حذف می‌شوند.

۷- نتیجه‌گیری

با توجه به پیشینه کار و با توجه به نیاز شدید توسعه‌دهندگان برای انتخاب یک روش رسمی مناسب از میان چندین روش، نتایج به‌دست آمده در جدول راهبرد به‌عنوان مرجع استفاده می‌شود. یک توسعه‌دهنده نرم‌افزار می‌تواند با مراجعه به جدول راهبرد، روش رسمی را انتخاب کند که با نرم‌افزار تحت توسعه خود متناسب باشد. علاوه بر آن می‌تواند با انتخاب روش رسمی مناسب از این جدول، با استفاده از جدول برازندگی، از ویژگی‌ها، ضعف‌ها و قابلیت‌های روش رسمی انتخابی آگاه شود.

نتایج به‌دست آمده از این تحقیق از دو منظر می‌تواند مورد استفاده قرار گیرد:

۱) از منظر کاربر: کاربر با اتکا به معیارها، روش یا روش‌های مناسبی را برای توصیف و تحلیل انتخاب نماید. انتخاب یک روش یا ترکیب چند روش مناسب با توجه به معیارها، کمک شایانی جهت توصیف دقیق و تحلیل نرم‌افزار خواهد کرد.

۲) از منظر محققان: معیارها به محققان و توسعه‌دهندگان روش‌های رسمی جهت فهم بهتر و بیشتر روش‌ها و همچنین اصول پایه‌ای روش‌ها کمک نموده و موجب می‌شود تا روش‌ها بهبود پیدا کنند یا روش‌های جدید ترکیبی ارائه گردند و نهایتاً این امکان را برای محققان فراهم می‌آورد تا یک روش مرجع را توسعه داده که این روش مرجع می‌تواند برای شناسایی وجوه تشابه و تمایز بین روش‌های مختلف استفاده شود.

یک توسعه‌دهنده نرم‌افزار با استفاده از ارزیابی‌های انجام‌گرفته بر روی روش‌های رسمی، به راحتی می‌تواند روشی متناسب با نرم‌افزار تحت توسعه خود انتخاب نماید که انتخاب روش مناسب برای توصیف و تحلیل نرم‌افزار می‌تواند مزایای زیر را به همراه داشته باشد:

۱) توصیف و تحلیل نرم‌افزار با صرف کمترین زمان.

۲) افزایش توانایی توصیف و تحلیل نیازمندی‌های نرم‌افزار و نهایتاً افزایش دقت در توصیف و تحلیل آنها.

۳) افزایش قابلیت اطمینان و کارآیی نرم‌افزار.

تحقیق انجام‌شده در این مقاله می‌تواند با ارزیابی و تحلیل روش‌های رسمی بیشتر، در تحقیقات آتی توسعه یابد و نتایج این ارزیابی‌ها به‌صورت جداول برازندگی و راهبرد در اختیار توسعه‌دهندگان نرم‌افزار قرار گیرد. علاوه بر این می‌بایست بر اساس ضعف‌های هر روش تلاش نمود تا روش‌های دارای ضعف، توسعه داده شوند یا جهت تکمیل با سایر روش‌ها تلفیق گردند.

مراجع

[1] E. M. Clarke and J. M. Wing, "Formal methods: state of the art and future directions," *ACM Computing Surveys*, vol. 28, no. 4, pp. 626-643, Dec. 1996.

[2] C. Heitmeyer, "On the need for practical formal methods," in Formal Techniques in Real Time and Real - Time Fault - Tolerant Systems, Proc. of the 5th Intern. Symposium, Springer, pp. 18-26, 1998.

[3] H. Oberheid, "A coloured Petri - Net model of cooperative arrival planning in air traffic control," in *K. Jensen (Ed.): Proc. of the 17th Worskhop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools*, pp. 177-196, 2006.

[4] C. Heitmeyer, M. Archer, R. Bharadwaj, and R. Jeffords, "Tools for constructing requirements specifications, the SCR toolset at the age of ten," *Int. J. of Comput. Syst. Sci. & Eng.*, vol. 1, pp. 19-35, 2005.

[5] B. Berard, F. Cassez, S. Haddad, D. Lime, and O. H. Roux, "Comparison of the expressiveness of timed automata and time Petri Nets," in *Proc. of 3rd Int. Conf. on Formal Modeling and Analysis of Timed Systems*, vol. 3829, pp. 211-225, 2005.

[6] H. Habrias and M. Frrapier, "Software specification methods, an overview using a case study," Book Chapter in *Formal Approach to Computing and Information Technology*, pp. 11-20, 2006.

[7] I. V. Weerd, S. Weerd, and S. Brinkkemper, "Developing a reference method for game production by method comparison," *in Proc. of IFIP Int. Federation for Information Processing, Spring*, vol. 244, pp. 327-313, 2007.

[8] X. Hei, L. Chang, W. Ma, J. Gao, and G. Xie, "Automatic transformation from UML Statechart to Petri Nets for safety analysis and verification," in *Proc. of Int Conf. on Quality, Reliability, Risk, Maintenance, and Safety Engineering*, pp. 948-951, 2011.

[9] A. M. K. Cheng, Real - Time Systems: Scheduling, Analysis, and Verification, Wiley, 2002.

[10] R. M. Amadio and D. Lugiez (Eds), *Concurrency Theory*, Springer, 2003.

[11] G. Sukumar, Distributed Systems - an Algorithmic Approach, Chapman & Hall/CRC, 2007.

[12] I. Fogg, B. Hicks, A. Lister, T. Mansfield, and K. Reymond, "A comparison of Lotos and Z for specifying distributed systems," *Australian Computer Science Communications*, vol. 12, no. 1, pp. 88-96, Feb. 1990.

[13] R. Stuart and P. Norvig, *Artificial Intelligence: a Modern Approach*, 3rd Ed., Prentice Hall, 2010.

[14] K. Munn and B. Smith, *Applied Ontology: an Introduction*, Ontos Verlag, 2008.

[15] M. Jarrar and R. Meersman, "Ontology engineering - the DOGMA approach," *Advances in Web Semantics I, LNCS*, vol. 4891, pp. 7-34, Springer. 2008.
[16] B. Smith et al, "The OBO foundry: coordinated evolution of ontologies to support biomedical data integration," *Journal of Nature Biotechnology*, vol. 25, no. 11, pp. 1251-1255, 2007.
[17] P. McCorduck, *Machines Who Think*, 2nd Ed., A K Peters/CRC Press, 2004.
[18] I. Brakto, Prolog Programming for Artificial Intelligence, Addison Wesley, 2001.
[19] J. C. Giarratano and R. Gary, *Expert Systems, Principles and Programming*, Thomson Course Technolog, 2005.
[20] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, no. 3, pp. 338-353, 1965.
[21] P. J. Haas, "Stochastic Petri Nets: modeling, stability, simulation," Springer, 2002.
[22] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel - Schneider, *The Description Logic Handbook*, Cambridge University Press, 2007.
[23] S. M. Brien, "The development of Z., In D. J. Andrews, J. F. Groote, and C. A. Middelburg, editors, Semantics of Specification Languages (SoSL), Workshops in Computing, pp. 1-14, Springer, 1994.
[24] J. P. Bowen, *Formal Specification and Documentation Using Z: A Case Study Approach*, International Thomson Computer Press, pp. 1-65, 2003.
[25] S. Stepney, R. Barden, and D. Cooper (Eds.), "Object orientation in Z," *Workshops in Computing, Springer*, 1992.
[26] http://czt.sourceforge.net/index.html
[27] J. R. Abrial, Modeling in Event - B: System and Software Engineering, Cambridge University Press, 2010.
[28] http://www.event-b.org/platform.html
[29] F. Jahanian and A. Mok, "Modechart: a specification language for real - time systems," *IEEE Trans. on Software Engineering*, vol. 20, no. 12, pp. 933-947, Dec. 1994.
[30] D. A. Stuart, "Implementing a verifier for real - time systems," in *Proc. of IEEE Real - Time Systems Symposium*, pp. 62-71, 1990.
[31] A. E. Abdallah, C. B. Jones, and J. W. Sanders (Eds.), "Communicating sequential processes, the first 25 years," *Lecture Notes in Computer Science*, vol. 3525, 2005.
[32] J. Carter and W. B. Gardner, "A formal CSP framework for message - passing HPC programming," in *Proc. of Canadian Conf. on Electrical and Computer Engineering*, pp. 1466-1470, 2006.
[33] S. Jun, Y. Liu, and J. Song Dong, "Model checking CSP revisited: introducing a process analysis toolkit," in *Proc. of the 3rd International Symposium on Leveraging Applications of Formal Methods, Verification, and Validation, Springer*, pp. 307-322, 2008.

سید مرتضی بابامیر تحصیلات خود را در مقطع کارشناسی مهندسی نرم‌افزار از دانشگاه فردوسی مشهد و تحصیلات خود را در مقاطع کارشناسی ارشد و دکتری مهندسی نرم‌افزار در سال‌ای ۱۳۷۹ و ۱۳۸۶ از دانشگاه تربیت مدرس به پایان رساند و هم‌اکنون استادیار دانشکده مهندسی برق و کامپیوتر دانشگاه کاشان است. زمینه‌های تحقیقاتی مورد علاقه ایشان عبارتند از: مهندسی نرم‌افزار، روش‌های رسمی، توصیف و آزمون نرم‌افزار، سیستم‌های توزیعی و شی‌گرا و زبان‌های برنامه‌سازی.

ویدا احمدی ثابت تحصیلات خود را در مقطع کارشناسی مهندسی نرم افزار در سال ۱۳۸۳ از دانشگاه بوعلی سینا- همدان و تحصیلات خود را در مقطع کارشناسی ارشد مهندسی نرم‌افزار در سال ۱۳۸۹ از دانشگاه آزاد اسلامی اراک به پایان رساند و هم‌اکنون مربی دانشگاه پیام نور- واحد همدان است. زمینه‌های تحقیقاتی مورد علاقه ایشان عبارتند از: مهندسی نرم‌افزار، روش‌های رسمی، سیستم‌های هوشمند.