دانشکدگان علوم
دانشکده ریاضی، آمار و علوم کامپیوتر

بهبود روش وارسی مدل با استفاده از تعبیر مجرد

نگارنده

پویا پرتو

استاد راهنمای اول: دکتر مجید علی‌زاده
استاد راهنمای دوم: دکتر مجتبی مجتهدی

پایاننامه برای دریافت درجه کارشناسی ارشد
در رشته علوم کامپیوتر

تاریخ دفاع

چکیده

روش وارسی مدل یک روش قابل اعتماد برای بررسی صحت عملکرد برنامه‌های کامپیوتری است. بیان‌های مختلف این روش از منطق موجهات بهره می‌برند که چندان برای برنامه نویسان شناخته شده نیستند. در این رساله سعی شده است که یک بیان جدید از روش وارسی مدل مورد شرح و بررسی قرار گیرد که در ادبیات نظریه تعبیر مجرد بیان شده است و در آن به جای منطق موجهات از عبارات منظم استفاده شده است.

پس از ارائه‌ی مفاهیم اولیه، به سه صورت متفاوت به بیانی جدید از روش وارسی مدل پرداخته‌ایم. صورت اول ساختار خاصی ندارد و صرفا در ادبیات نو بیان شده است، صورت دوم ساختار عبارات منظم را به صورتبندی‌اش اضافه کرده است و در صورت سوم، با اضافه شدن ساختار برنامه به صورتبندی، روش به پیاده سازی نزدیک‌تر شده است. معادل بودن این سه صورت نیز مطالعه و بررسی می‌شود.

کلمات کلیدی: وارسی مدل، نظریه تعبیر مجرد، معناشناسی دلالتی، درستی‌یابی صوری، تحلیل ایستا، درستی‌یابی برنامه‌های کامپیوتری

تقدیم به

تقدیم به

سپاسگزاری

سپاسگزاری

پیشگفتار

با توجه به پیشرفت روز افزون علوم کامپیوتر و ورود کاربردهای آن به زندگی روزمره، پیشرفت در روش‌های ساخت و نگه‌داری برنامه‌ها نیازی آشکار به نظر می‌رسد. یکی از مسائل مهم در این زمینه بررسی صحت کارکرد برنامه‌های نوشته‌شده است. عدم صحت کارکرد برنامه‌های نوشته شده بسته به حساسیت یک برنامه می‌تواند تبعات زیان‌بار جبران ناپذیری به همراه داشته‌باشد. پرتاب ناموفق آریان ۵[۱۸] ، از مدار خارج شدن مدارگرد مریخ [۲] و تصادف هلیکوپتر چینوک [۱] چند نمونه از تبعات بزرگ این قضیه در گذشته بوده اند، همین طور به‌سادگی می‌توان فجایع دیگری از این دست را در زندگی روزمره‌ی انسان‌ها متصور شد.

برای تعیین صحت کارکرد برنامه‌های کامپیوتری روش های متفاوتی ابداع شده‌اند که در ادامه به طور مختصر از آن‌ها یاد می‌کنیم، اما پیش از آن به یک خاصیت مشترک همه‌ی این روش‌ها، یعنی "ناکامل بودن"، می‌پردازیم. منظور از ناکامل بودن این است که با استفاده از هیچ یک از روش‌هایی که داریم، نمی‌توانیم هر خاصیتی را برای هر برنامه‌ای بررسی کنیم. به عبارت دیگر، استفاده از هر روشی محدودیت‌هایی دارد. البته قضیه رایس [۲۲] به ما این تضمین را داده که روش کاملی اصلا وجود ندارد. قضیه رایس(به طور غیر رسمی) بیان می‌کند که مسئله ی بررسی هر خاصیت غیر بدیهی، برای همه ی برنامه‌ها، تصمیم ناپذیر است. این دلیلی بر این شده که روش‌های مختلفی برای این کار معرفی شوند که هر کدام می‌توانند حالت‌های خاصی از مسئله را حل کنند.

یک دسته‌بندی برای این روش‌ها تقسیم آن‌ها به دو دسته‌ی پویا و ایستا است. روش‌های پویا روش‌هایی هستند که در آن‌ها تست برنامه همزمان با اجرای برنامه است، درحالیکه روش‌های ایستا بدون اجرای برنامه‌ها آن‌ها را تست می‌کنند.

روش‌های پویا معمولاً با اجرای حالات محدودی از برنامه تصمیم می‌گیرند که برنامه‌ای که نوشته شده است، انتظارات را برآورده می‌کند یا خیر. اگر این روش بتواند تشخیص دهد که برنامه‌ای درست کار نمی‌کند، می‌توانیم با اطمینان نتیجه بگیریم که آن برنامه غلط نوشته‌شده است، اما اگر

برنامه‌ای از تست‌های ساخته‌شده با این روش‌ها با موفقیت عبور کند، نمی‌توان اطمینان حاصل کرد که برنامه درست کار می‌کند، زیرا ممکن است، حالتی مشکل‌زا از اجرای برنامه وجود داشته باشد که در تست ها نیامده باشد. برای اطلاعات بیشتر به [۲۰] مراجعه کنید.

روش‌های ایستا معمولاً روش‌هایی هستند که از نظریه‌های مختلف در منطق ریاضی به عنوان ابزار بهره می‌برند تا بدون اجرای خود برنامه‌ها در مورد صحت اجرای آن‌ها نتیجه‌گیری کنند. به همین دلیل به بخشی مهم و بزرگی از این دستورات که از منطق استفاده می‌کنند روش‌های صوری هم گفته‌می‌شود. معروف ترین روش‌های ایستا؛ روش وارسی مدل، روش‌های استنتاجی و استفاده از نظریه تعبیر مجرد است.

در روش وارسی مدل، یک مدل صوری متناهی از برنامه‌ی موردبررسی می‌سازیم که همه‌ی حالات اجرای برنامه با آن قابل‌توصیف است، سپس با استفاده از یک زبان صوری که بتواند در مورد مدل هایمان صحبت‌کند، ویژگی‌های مورد بررسی را بیان‌می‌کنیم و در نهایت صحت ویژگی‌های بیان‌شده را بررسی می کنیم. مقاله [۴] شروع این روش‌ها بوده که این کار را با استفاده از نوعی مدل کریپکی [۱۷] و نوعی منطق زمانی به نام منطق زمانی خطی [۴] انجام داده که روشی است با دقت و البته هزینه‌ی محاسباتی بسیار بالا. [۱۲] یک منبع بسیار مقدماتی و کتاب[۵] یک مرجع سنتی در این زمینه است.

در روش‌های استنتاجی که شاید بتوان یکی از ابتدایی ترین آن‌ها را استفاده از منطق هور[۱۱] دانست، درستی کارکرد برنامه‌هایمان را با ارائه‌ی یک درخت اثبات در یک دستگاه استنتاجی، متناسب با زبان برنامه‌هایمان، نشان می‌دهیم. در این روش هم اگر بتوانیم درستی یک برنامه را اثبات کنیم، دیگر به طور نظری، خیالی آسوده از درستی برنامه خواهیم داشت، اما ساختن درخت اثبات در یک نظریه برهان می‌تواند چالش برانگیز باشد. در[۱۲] به منطق هور به طور مقدماتی پرداخته شده است. همین طور کتاب[۲۱] نیز به پیاده‌سازی منطق هور در زبان coq پرداخته است، که در آن coq یک اثبات‌یار است که بر اساس نظریه نوع وابسته کار می‌کند. برای اطلاعات بیشتر در مورد چگونگی طرز کار این اثبات‌یار و نظریه‌ی بنیادین آن به کتاب[۳] مراجعه کنید. نظریه‌ی مورد شرح در[۱۰] نیز می‌تواند در این مسیر به کار گرفته شود.

نظریه تعبیر مجرد[۸] نیز یک نظریه ریاضیاتی است که به‌نوعی سعی می‌کند از روی معناشناسی یک برنامه‌ی کامپیوتری[۲۴] یک تقریب بسازد. منظور از تقریب یک دستگاه کوچک‌تر از معناشناسی اصلی است که رفتارش زیرمجموعه‌ی رفتارهای دستگاه اصلی است. سعی بر این است که دستگاه جدیدی که می‌سازیم به لحاظ محاسباتی ساده‌تر از معناشناسی اصلی کار کند تا بتوان خواص آن را راحت‌تر بررسی کرد. در این صورت هر نتیجه‌ای در مورد خواص جدید، را می‌توان

برای خود برنامه هم بیان کرد، اما توجه شود که در این صورت ممکن است به همه‌ی حقایق دست پیدا نکنیم. برای اطلاعات بیشتر به [۷] و [۱۴] مراجعه شود.

فهرست مطالب

فصل ۱

مقدمه و مفاهیم اولیه

در این فصل به عنوان مقدمه، روش وارسی مدل به طور مختصر معرفی شده‌است. در فصل‌های بعدی، با هدف بهبود این روش، صورت‌های جدیدی از آن ارائه شده و مورد بررسی قرار گرفته است، بنابراین، لازم است که ابتدا، به معرفی این روش به شکل سنتی پرداخته شود.

پس از معرفی وارسی مدل، بحث اصلی این پایان نامه شروع می‌شود. محوریت کار ما[۶] است که در آن روش جدید وارسی مدل ارائه شده است. بحث با ارائه‌ی نحو یک زبان برنامه نویسی شروع می‌شود، سپس معناشناسی این زبان، یعنی معناشناسی رد پیشوندی[1] ارائه می‌شود و فصل تمام می‌شود. مفاهیم معرفی شده در این فصل دارای ریزه‌کاری‌های زیادی هستند و به عقیده‌ی نگارنده، در [۶] در ارائه‌ی بعضی از جزئیات سهل انگاری اتفاق افتاده است. سعی کرده‌ایم که اگر ایرادی در تعاریف موجود در [۶] وجود دارد را حین بیان دوباره‌ی این مفاهیم در این پایان نامه رفع کنیم، تا یک بیان خوش ساخت و روان از این نظریه ارائه کرده باشیم.

۱.۱ روش وارسی مدل

روش وارسی مدل یک روش صوری است که برای درستی‌یابی سیستم‌های مختلف استفاده می‌شود. در این روش معمولا ابتدا یک ماشین حالات متناهی از روی سیستم مورد بررسی ساخته می‌شود، سپس بررسی‌هایی که قرار است روی سیستم اصلی انجام شوند، روی این مدل انجام می‌شود.

[1] Prefix Trace Semantics

از این روش در بررسی صحت کارکرد برنامه‌های کامپیوتری استفاده می‌شود، اما این تنها مورد استفاده‌ی این روش نیست. هر سیستمی که قابلیت بیان شدن به طور صوری را داشته باشد، با این روش قابل بررسی است. مثلا می‌توان از این روش برای بررسی صحت عملکرد یک برنامه برای قطارهای شهری استفاده کرد. در یک برنامه برای قطارهای شهری، نباید امکان حضور دو قطار روی یک ریل در یک زمان وجود داشته باشد(که معنی تصادف بین دو قطار را می‌دهد) و می‌توان از روش وارسی مدل برای اطمینان از عدم وجود چنین ویژگی نامطلوبی استفاده کرد. مثال های دیگر استفاده‌ی این روش در علوم کامپیوتر بررسی صحت عملکرد معماری یک پردازنده یا الگوریتم زمانبندی یک سیستم عامل است. این مثال‌ها هیچ کدام یک برنامه‌ی کامپیوتری نیستند(هر چند که ممکن است مجبور باشیم از یک برنامه‌ی کامپیوتری برای پیاده‌سازی آن‌ها کمک بگیریم که در آن صورت بررسی صحت عملکرد آن برنامه‌ی کامپیوتری داستانی دیگر خواهد داشت)، اما قابل بیان به صورت صوری به جای زبان طبیعی هستند.

روش وارسی مدل برای بیان ویژگی‌های مورد بررسی از منطق‌های زمانی مختلف استفاده می‌کند. منطق زمانی یک نوع منطق موجهات است. منطق‌های موجهات از گسترش زبان منطق کلاسیک، با اضافه کردن ادوات وجهی گوناگون، ساخته می‌شوند. این ادوات غالبا در زبان طبیعی نقش قید را دارند. منطق‌های زمانی دسته‌ای از منطق‌های موجهات هستند که به صوری‌گری ما مفهوم زمان را اضافه می‌کنند، یعنی قیدهایی مانند فعلا، بعدا، و قبلا(که مورد آخری کمتر رایج است). منطقی که در اینجا بیان می‌کنیم منطق زمانی خطی[۲] یا LTL نام دارد که یکی از منطق‌های زمانی است که برای روش وارسی مدل استفاده می‌شود. البته در مورد قیدهای مذکور، اشاره به این نکته ضروری است که در بیانی که در اینجا از این منطق ارائه داده‌ایم، ادوات جدید به طور مستقیم بیانگر این قیدها نیستند، هرچند که به کمک ادوات جدید می‌توان ادواتی برای هر یک از این قیدها ساخت. این تعاریف از [۱۹] آورده شده‌اند.

ابتدا نحو این منطق را بیان می‌کنیم و سعی می‌کنیم، به طور غیر دقیق، در مورد معنای فرمول‌های این زبان به خواننده یک درک شهودی بدهیم، سپس به سراغ معناشناسی صوری این منطق می‌رویم.

[2] Linear Temporal Logic

۱.۱.۱ زبان LTL

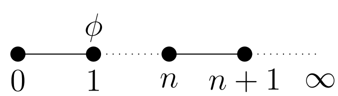
تعریف ۱.۱. هر عضو مجموعه‌ی Φ یک فرمول در زبان LTL است و Π مجموعه‌ی(شمارای نامتناهی) فرمول‌های اتمی است و $\pi \in \Pi$:

$$\Pi \subset \Phi,$$

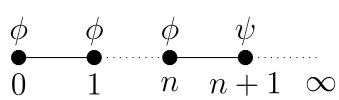
$$\phi \in \Phi \Leftrightarrow \phi ::= \pi | \phi \vee \phi | \neg\phi | \bigcirc \phi | \phi \mathcal{U} \phi$$

در این منطق، زمان را با اعداد طبیعی نشان می‌دهیم. یعنی برای یک فرمول، زمان از عدد ۰ شروع شده و تا ابد ادامه خواهد داشت و حین گذر زمان ممکن است ارزش فرمول‌ها تغییر کند. مسلما پس از بررسی معناشناسی صوری بهتر می‌شود این مفهوم را به طور شهودی حس کرد، اما به هر حال به خواننده پیشنهاد می‌شود، پیش از رسیدن به آن بخش به ادامه‌ی این بخش که در تلاش است یک درک شهودی از معنای فرمول‌ها بدهد، توجه کند.

در این زبان ادوات کلاسیک $\vee$,$\neg$ هستند، به همان معنایی که در منطق گزاره‌ای کلاسیک داشتند. در ادوات جدید $\bigcirc\phi$ به معنای برقرار بودن این فرمول دقیقا در لحظه‌ی بعدی(دقیقا یک لحظه) است، مثلا در شکل زیر با در نظر گرفتن اینکه در زمان ۰ هستیم، این فرمول در لحظه‌ی ۱ برقرار است.



$\phi \mathcal{U} \psi$ به این معنی است که فرمول سمت چپی حداقل تا قبل از اینکه فرمول سمت راستی برقرار شود، برقرار است.(مثلا اگر بگوییم "تا وقتی که باران نباریده زمین خشک است" در این صورت "زمین خشک است" به جای فرمول سمت چپ و "باران باریده است" فرمول سمت راست است).



این زبان را می‌توان با ادوات بیشتری از آنچه آورده‌ایم بیان کرد و البته بیان‌های دیگری هم بسته به بحث متداول هستند، اما در اینجا یک شکل ساده از این زبان را آورده‌ایم که به غیر از ادوات منطق گزاره‌ای کلاسیک دو ادات دیگر را در زبان خود دارد. دلیل وجود ادوات متفاوت، می‌تواند راحت‌تر کردن بیان ویژگی‌ها باشد. همان طور که استفاده نکردن از فصل و شرطی در منطق

گزاره‌ای کلاسیک می‌تواند، به سخت کردن بیان جملات در چارچوب این منطق منجر شود، حذف این ادوات وجهی هم بیان ویژگی‌ها را در این منطق مشکل می‌سازد.

حال که به درکی شهودی از معنای فرمول‌های این زبان رسیده‌ایم، به بیان صوری این مفاهیم می‌پردازیم.

۲.۱.۱ معناشناسی LTL

مدل‌های این منطق را به صورت توابع $M : \mathbb{N}_0 \to P(\Pi)$ تعریف می‌کنیم. به عبارت دیگر، هر مدل یک تابع است که هر عدد طبیعی را به یک مجموعه از فرمول‌های اتمی می‌برد. این در واقع به این معنی است که یک مدل مشخص می‌کند که در هر لحظه کدام یک از فرمول‌های اتمی درست هستند. مثلا، در مدلی به نام M در واقع $M(5)$ مجموعه‌ی اتم‌هایی است که در لحظه‌ی ۵ طبق این مدل درست هستند و اگر اتمی در این مجموعه حاضر نباشد، در لحظه‌ی ۵، ارزش غلط دارد. درستی یک فرمول در یک مدل را با M, i نشان می‌دهیم و $M, i \models \phi$ به این معنی است که فرمول ϕ، در لحظه‌ی iدر مدل M درست است. این مفهوم را، به صورت بازگشتی، به شکل زیر تعریف می‌کنیم:

$M, i \models \pi \quad iff \quad \pi \in M(i)$
$M, i \models \neg\phi \quad iff \quad M, i \nvDash \phi$
$M, i \models \phi \vee \psi \quad iff \quad M, i \models \phi \quad or \quad M, i \models \psi$
$M, i \models \bigcirc\phi \quad iff \quad M, i+1 \models \phi$
$M, i \models \phi\mathcal{U}\psi \quad iff \quad \exists k \geq i \in \mathbb{N}_0 : \forall i \leq j < k : M, j \models \phi \quad and \quad M, k \models \psi$

یک فرمول را ارضاپذیر می‌گوییم اگر و تنها اگر مدلی وجود داشته باشد که فرمول در آن درست باشد. اگر یک فرمول در هر مدلی درست باشد، آن فرمول را معتبر می‌گوییم.

۲.۱ نحو زبان مورد بررسی

نحو زبان بیان برنامه‌ها زیرمجموعه‌ای از نحو زبان C است، به شکل زیر:

$\mathsf{x}, \mathsf{y}, \ldots \in \mathbb{X}$

$$
\begin{aligned}
&\mathsf{A} \in \mathbb{A} ::= 1 \mid \mathsf{x} \mid \mathsf{A}_1 - \mathsf{A}_2 \\
&\mathsf{B} \in \mathbb{B} ::= \mathsf{A}_1 < \mathsf{A}_2 \mid \mathsf{B}_1 \ \mathsf{nand} \ \mathsf{B}_2 \\
&\mathsf{E} \in \mathbb{E} ::= \mathsf{A} \mid \mathsf{B} \\
&\mathsf{S} \in \mathbb{S} ::= \\
&\qquad\quad \mathsf{x} \doteq \mathsf{A}; \\
&\qquad \mid \ ; \\
&\qquad \mid \ \mathsf{if}\ (\mathsf{B})\ \mathsf{S} \mid \mathsf{if}\ (\mathsf{B})\ \mathsf{S}\ \mathsf{else}\ \mathsf{S} \\
&\qquad \mid \ \mathsf{while}\ (\mathsf{B})\ \mathsf{S} \mid \mathsf{break}; \\
&\qquad \mid \ \{\mathsf{Sl}\} \\
&\mathsf{Sl} \in \mathbb{SL} ::= \mathsf{Sl}\ \ \mathsf{S} \mid \epsilon \\
&\mathsf{P} \in \mathbb{P} ::= \mathsf{Sl}
\end{aligned}
$$

قابل مشاهده است که این زبان، نسبت به کل زبان C، تا حد ممکن ساده‌سازی شده است. علت این کار را بعدا عمیق‌تر حس خواهیم‌کرد. علتْ ساده‌تر شدن کار برای ارائه‌ی معناشناسی و صورت‌های جدید روش وارسی مدل است. در اینجا، راحتی برنامه نوشتن در این زبان مطرح نبوده است، چون اصلا این زبان برای این کار ساخته نشده است. هدف از ارائه‌ی این زبان صرفا ارائه‌ی روش جدید است. یعنی می‌توان به این زبان به چشم یک مدل محاسبه، مانند ماشین تورینگ[3] و ماشین رجیستر[4] نگاه کرد. روشی که سعی در ارائه‌ی آن داریم، برای زبان‌های برنامه نویسی دستوری است، مانند پایتون[5]، جاوا[6] و C. بنابراین، انتخاب یک مدل محاسبه که رفتاری شبیه‌تر به این زبان‌ها داشته باشد، کار معقولی است.

اندکی در مورد قدرت بیان این زبان صحبت می‌کنیم. می‌توانیم باقی اعداد را از روی عدد ۱ و عملگر منها بسازیم. مثلا ابتدا ۰ را به کمک ۱ ـ ۱ می سازیم و سپس، با استفاده از ۰ می‌توانیم یکی

[3] Turing Machine
[4] Register Machine
[5] Python
[6] Java

یکی اعداد صحیح منفی را بسازیم. پس از آن، به سراغ اعداد صحیح مثبت می‌رویم که با کمک ۰ و هر عدد صحیح منفی‌ای که ساخته‌ایم، ساخته می‌شوند. باقی اعداد و حتی عملگرها(یعنی به غیر از اعداد طبیعی) از روی اعداد و عملگرهایی که داریم قابل‌ساختن هستند. در عبارت‌های بولی[7] نیز با داشتن دسته‌ای محدود اما کافی از عملگرها، باقی عملگرهای رایج قابل بیان هستند. یعنی اینجا صرفا ادات شفر[8] تعریف شده است و باقی عملگرهای بولی[9]را می‌توان با استفاده از همین عملگر ساخت. باقی دستورات نیز دستورات شرط و حلقه هستند. باقی دستورات مطابق رفتاری که از آن‌ها در زبان C انتظار داریم کار می‌کنند. در مورد دستور ;break ذکر این نکته ضروری است که اجرای این دستور، اجرای برنامه را از دستوری بعد از داخلی‌ترین حلقه‌ای که ;break داخلش قرار دارد، ادامه می‌دهد. در پایان می توان ثابت کرد که این زبان هم‌ارز با ماشین تورینگ[۹] است.

هرآنچه بالاتر درمورد معنای دستورات این زبان گفتیم، به هیچ وجه صوری نبود. صرفا یک درک شهودی که از معنای اجرای هریک از دستورات می‌توان داشت را بیان کرده‌ایم. بیان صوری معنای برنامه‌ها را که قابل انتقال به کامپیوتر است(خلاف درک شهودیمان)، در ادامه بیان خواهیم‌کرد. بدیهی است که یک بیان صوری از روی یک درک شهودی ساخته می‌شود، اما این دو استفاده‌های متفاوتی دارند.

۳.۱ معناشناسی زبان مورد بررسی

معناشناسی زبانی را که در بخش پیش آورده‌ایم، با کمک مفاهیمی به نام‌های برچسب[10] و رد پیشوندی[11] و عملگری به نام چسباندن[12] تعریف می‌کنیم. نام این معناشناسی، معناشناسی رد پیشوندی است.

---

[7]Boolean Expression
[8]Sheffer Operator
[9]Boolean Operator
[10]Label
[11]Prefix Trace
[12]Concatenation

۱.۳.۱ برچسب‌ها

باوجود اینکه در نحو زبان C برچسب‌ها وجود دارند،اما در نحو زبانی که معرفی کرده‌ایم، برچسب‌ها حضور ندارند. با این وجود، در تعریف معناشناسی صوری برای برنامه‌ها، به این مفهوم نیاز داریم. در این بخش، به طور غیر دقیق معنای برچسب‌ها را آورده‌ایم. همین تعاریف غیر دقیق برای کار ما کافی است. تعاریف صوری دقیق‌تر برچسب‌ها در پیوست [۶] آورده‌شده‌اند. از آوردن مستقیم این تعاریف در اینجا خودداری کرده‌ایم. البته در مورد تعریف صوری برجسب‌ها، قابل ذکر است که طبق [۷]، تعریف صوری برچسب‌ها غیر قطعی است(به عبارتی منفی نگرانه خوش تعریف نیست). به عبارت دیگر، تعریف‌های صوری مشخص‌تر متفاوتی را می‌توان متصور شد که در تعریف صوری برچسب‌ها در [۷] می‌گنجند.

در نحو زبانمان، Sها بخشی از عبارت‌های موجود در نحو زبان هستند که به آن‌ها عبارت دستوری[۱۳] می‌گوییم. برچسب‌ها را برای Sها تعریف می‌کنیم. برچسب‌ها با کمک توابع labs, in, brks-of, brk-to, esc, aft, at تعریف می‌شوند. در واقع، هر S، به ازای هر یک از این توابع، ممکن است یک برچسب متفاوت داشته باشد. بعضی دیگر از این توابع به ازای هر S ممکن است یک مجموعه از برچسب‌ها را برگردانند. یکی از آن‌ها هم با گرفتن S یک مقدار بولی[۱۴] را برمی‌گرداند.

$\text{at}[\![S]\!]$: برچسب شروع S.

$\text{aft}[\![S]\!]$: برچسب پایان S، اگر پایانی داشته باشد.

$\text{esc}[\![S]\!]$: یک مقدار بولی را بازمی‌گرداند که بسته به اینکه در S عبارت دستوری ;break وجود دارد یا خیر، مقدار درست یا غلط را برمی‌گرداند.

$\text{brk-to}[\![S]\!]$: برچسبی است که اگر حین اجرای S عبارت دستوری ;break اجرا شود، برنامه از آن برچسب ادامه پیدا می کند.

$\text{brks-of}[\![S]\!]$: مجموعه‌ای از برچسب‌های عبارت‌های دستوری ;break که داخل S هستند را برمی‌گرداند.

$\text{in}[\![S]\!]$: مجموعه‌ای از تمام برچسب‌های درون S را برمی‌گرداند.

$\text{labs}[\![S]\!]$: مجموعه‌ای از تمام برچسب‌هایی که با اجرای S قابل دسترسی هستند را برمی‌گرداند.

---
[13] Statement

[14] Boolean Value

مجموعه‌ی همه‌ی برچسب‌ها را با $\mathbb{L}$ نشان می‌دهیم.

۲.۳.۱ رد پیشوندی

حال که تعریف برچسب‌ها را داریم، به سراغ تعریف رد پیشوندی می‌رویم. البته پیش از آن، باید وضعیت[۱۵] و محیط[۱۶] را تعریف کنیم.

تعریف ۲.۱. (محیط): به ازای مجموعه مقادیر[۱۷] $\mathbb{V}$ و مجموعه متغیرها[۱۸] $\mathbb{X}$ تابع $\rho : \mathbb{X} \to \mathbb{V}$ را یک محیط می‌گوییم. مجموعه‌ی همه‌ی محیط‌ها را با $\mathbb{EV}$ نمایش می‌دهیم.

تعریف ۳.۱. (وضعیت): به هر زوج مرتب متشکل از یک برچسب l و یک محیط ρ یک وضعیت $\langle l, \rho \rangle$ می‌گوییم. مجموعه‌ی همه‌ی وضعیت‌ها را با $\mathfrak{S}$ نشان می‌دهیم.

تعریف ۴.۱. (رد پیشوندی): به یک دنباله از وضعیت‌ها(با امکان تهی بودن) یک رد پیشوندی می‌گوییم.

هر رد پیشوندی یک دنباله است که قرار است توصیفی از گردش کار[۱۹] اجرای یک برنامه باشد. منظور از گردش کار این است که در هر لحظه، حافظه چه وضعیتی دارد و اینکه برنامه در حال اجرای چه دستوری است. وضعیت‌ها موقعیت لحظه‌ای حافظه‌ای که در دسترس برنامه است را توصیف می‌کنند. l برچسب قسمتی از برنامه است که در حال اجرا است و ρ مقدار متغیرها را در آن لحظه از اجرای برنامه نشان می‌دهد. ردهای پیشوندی می‌توانند متناهی یا نامتناهی باشند. مجموعه‌ی ردهای پیشوندی متناهی را با $\mathfrak{S}^{+}$ و مجموعه‌ی ردهای پیشوندی نامتناهی را با $\mathfrak{S}^{\infty}$ نمایش می‌دهیم. مجموعه‌ی همه‌ی ردهای پیشوندی را هم با $\mathfrak{S}^{+\infty}$ نمایش می‌دهیم. با توجه به آنچه گفتیم، یک عملگر چسباندن $\bowtie$ را روی ردهای پیشوندی تعریف می‌کنیم.

پیش از ارائه‌ی تعریف، به دو نکته‌ی مهم در مورد نمادگذاری‌های این پایان نامه اشاره می‌کنیم.

اولین نکته این است که حین ارائه‌ی تعریف‌ها، مانند تعریف عملگر چسباندن که در ادامه آمده است، اگر تعریف را روی یک ساختار یا با در نظر گرفتن پیش فرض‌های مختلف ارائه داده باشیم، هر فرض را با علامت ◀ نشان داده‌ایم. در اثبات‌ها، برای هر حالت از ▶ استفاده کرده‌ایم.

[15] State
[16] Environment
[17] Value
[18] Variable
[19] Workflow

نکته‌ی دوم در مورد نشان دادن ردهای پیشوندی است. اگر π_1, π_2 رد پیشوندی باشند و σ یک وضعیت باشد، $\pi_1\sigma$ به یک رد پیشوندی لزوما متناهی اشاره می‌کند که با وضعیت σ به پایان رسیده است، $\sigma\pi_1$ به یک رد پیشوندی اشاره می‌کند که با وضعیت σ شروع شده است و $\pi_1\pi_2$ به یک رد پیشوندی اشاره می‌کند که با π_1 شروع شده است و با π_2 ادامه پیدا می‌کند(π_1 باید متناهی باشد). توجه شود که $\pi_1\pi_2$ با چسباندن π_1 و π_2 $(\pi_1 \bowtie \pi_2)$به همدیگر متفاوت است. هنگامی که می‌گوییم رد پیشوندی π(یا موقعیت σ) به رد پیشوندی π' اضافه شده است، به رد پیشوندی $\pi'\pi$(یا $\pi'\sigma$) اشاره می‌کنیم.

تعریف ۵.۱. (عملگر چسباندن): اگر داشته باشیم $\pi_1, \pi_2 \in \mathfrak{S}^{+\infty}, \sigma_1, \sigma_2 \in \mathfrak{S}$ داریم:

$$\blacktriangleleft\ \pi_1 \in \mathfrak{S}^{\infty} :$$

$$\pi_1 \bowtie \pi_2 = \pi_1$$

$$\blacktriangleleft\ \pi_1 \in \mathfrak{S}^{+} :$$

$$\blacktriangleleft\blacktriangleleft\ \sigma_1 = \sigma_2 :$$

$$\pi_1\sigma_1 \bowtie \sigma_2\pi_2 = \pi_1\sigma_1\pi_2$$

$$\blacktriangleleft\blacktriangleleft\ \sigma_1 \neq \sigma_2 :$$

در این حالت $\pi_1 \bowtie \pi_2$ تعریف نمی‌شود.

همین طور، ϵ یک رد پیشوندی است که حاوی هیچ وضعیتی نیست. به عبارت دیگر، یک دنباله‌ی تهی است.

۳.۳.۱ معناشناسی رد پیشوندی

در این بخش، دو تابع $\mathcal{A}$ و $\mathcal{B}$ را به ترتیب روی عبارت‌های حسابی[۲۰] و عبارت‌های بولی، یعنی Aها و Bها تعریف می‌کنیم، سپس با کمک آنها تابع $\mathcal{S}^*$ را روی مجموعه‌ای از اجتماع معنای Sها و Slها تعریف می کنیم. به هر Sl لیست عبارت‌های دستوری[۲۱] می‌گوییم. پس در نهایت، هدف ما تعریف $\mathcal{S}^*$ است.

[20]Arithmetic Expression
[21]Statement List

تعریف ۶.۱. (معنای عبارت‌های حسابی ـ تابع $\mathcal{A}$): تابع $\mathcal{A} : \mathbb{A} \rightarrow \mathbb{EV} \rightarrow \mathbb{V}$ را به صورت بازگشتی روی ساختار $\mathsf{A} \in \mathbb{A}$ به شکل زیر تعریف می‌کنیم:

$$\blacktriangleleft \ \mathcal{A}[\![1]\!]\rho = 1$$

$$\blacktriangleleft \ \mathcal{A}[\![\mathsf{x}]\!]\rho = \rho(\mathsf{x})$$

$$\blacktriangleleft \ \mathcal{A}[\![\mathsf{A}_1 - \mathsf{A}_2]\!]\rho = \mathcal{A}[\![\mathsf{A}_1]\!]\rho - \mathcal{A}[\![\mathsf{A}_2]\!]\rho$$

تعریف ۷.۱. (معنای عبارت‌های بولی ـ تابع $\mathcal{B}$): تابع $\mathcal{B} : \mathbb{B} \rightarrow \mathbb{EV} \rightarrow \mathbb{BOOL}$ را به صورت بازگشتی روی ساختار $\mathsf{B} \in \mathbb{B}$ به شکل زیر تعریف می‌کنیم:

$\blacktriangleleft \ \mathcal{B}[\![\mathsf{A}_1 < \mathsf{A}_2]\!]\rho = True$ اگر $\mathcal{A}[\![\mathsf{A}_1]\!]\rho$ کوچکتر از $\mathcal{A}[\![\mathsf{A}_2]\!]\rho$ باشد

$\blacktriangleleft \ \mathcal{B}[\![\mathsf{A}_1 < \mathsf{A}_2]\!]\rho = False$ اگر $\mathcal{A}[\![\mathsf{A}_1]\!]\rho$ بزرگتر از $\mathcal{A}[\![\mathsf{A}_2]\!]\rho$ باشد

$$\blacktriangleleft \ \mathcal{B}[\![\mathsf{B}_1 \mathsf{nand} \mathsf{B}_2]\!]\rho = \neg(\mathcal{B}[\![\mathsf{B}_1]\!]\rho \wedge \mathcal{B}[\![\mathsf{B}_2]\!]\rho)$$

طبیعتا $\wedge$ و $\neg$ در فرازبان هستند.

در ادامه، به تعریف $\mathcal{S}^*$ می‌پردازیم. این کار را با تعریف $\mathcal{S}^*$ روی هر ساختار عبارت دستوری S، لیست عبارت دستوری Sl و برنامه‌ی P انجام می‌دهیم. یعنی دامنه‌ی $\mathcal{S}^*$ مجموعه‌ی $\mathbb{P} \cup \mathbb{Sl} \cup \mathbb{S}$ است. پیش از ادامه‌ی بحث، باید این نکته را در مورد علامت‌گذاری‌هایمان ذکر کنیم که منظور از $\mathsf{S} ::= l\mathsf{break};$ این است که تاکید کرده‌ایم که S با برچسب l شروع شده‌است، هرچند که همین طور که پیش‌تر گفته‌شد، l جزو زبان نیست.

تعریف ۸.۱. (معنای برنامه‌ها ـ تابع $\mathcal{S}^*$): اگر $\mathsf{S} ::= \mathsf{break};$ باشد، ردهای پیشوندی متناظر با اجرای این عبارت دستوری را به شکل مجموعه‌ی زیر تعریف می‌کنیم:

$$\mathcal{S}^*[\![\mathsf{S}]\!] = \{\langle at[\![\mathsf{S}]\!], \rho\rangle | \rho \in \mathbb{EV}\} \cup \{\langle at[\![\mathsf{S}]\!], \rho\rangle \langle brk-to[\![\mathsf{S}]\!], \rho\rangle | \rho \in \mathbb{EV}\}$$

اگر $\mathsf{S} ::= \mathsf{x} \doteq \mathsf{A};$ باشد، ردهای پیشوندی متناظر با اجرای این عبارت دستوری را به شکل مجموعه‌ی زیر تعریف می‌کنیم:

$$\mathcal{S}^*[\![\mathsf{S}]\!] = \{\langle at[\![\mathsf{S}]\!], \rho\rangle | \rho \in \mathbb{EV}\} \cup \{\langle at[\![\mathsf{S}]\!], \rho\rangle \langle aft[\![\mathsf{S}]\!], \rho[\mathsf{x} \leftarrow \mathcal{A}[\![\mathsf{A}]\!]\rho]\rangle | \rho \in \mathbb{EV}\}$$

اگر $\mathsf{S} ::= \mathsf{if(B)S_t}$ باشد، ردهای پیشوندی متناظر با اجرای این عبارت دستوری را به شکل مجموعه‌ی زیر تعریف می کنیم:

$$\mathcal{S}^*[\![\mathsf{S}]\!] = \{\langle at[\![\mathsf{S}]\!], \rho\rangle | \rho \in \mathbb{EV}\} \cup \{\langle at[\![\mathsf{S}]\!], \rho\rangle\langle aft[\![\mathsf{S}]\!], \rho\rangle | \mathcal{B}[\![\mathsf{B}]\!]\rho = False\}$$

$$\cup\{\langle at[\![\mathsf{S}]\!], \rho\rangle\langle at[\![\mathsf{S_t}]\!], \rho\rangle\pi | \mathcal{B}[\![\mathsf{B}]\!]\rho = True \wedge \langle at[\![\mathsf{S_t}]\!], \rho\rangle\pi \in \mathcal{S}[\![\mathsf{S_t}]\!]\}$$

اگر $\mathsf{S} ::= \mathsf{if(B)S_t else S_f}$ باشد، ردهای پیشوندی متناظر با اجرای این عبارت دستوری را به شکل مجموعه‌ی زیر تعریف می کنیم:

$$\mathcal{S}[\![\mathsf{S}]\!] = \{\langle at[\![\mathsf{S}]\!], \rho\rangle | \rho \in \mathbb{EV}\}$$

$$\cup\{\langle at[\![\mathsf{S}]\!], \rho\rangle\langle at[\![\mathsf{S_f}]\!], \rho\rangle\pi | \mathcal{B}[\![\mathsf{B}]\!]\rho = False \wedge \langle at[\![\mathsf{S_f}]\!], \rho\rangle\pi \in \mathcal{S}[\![\mathsf{S_f}]\!]\}$$

$$\cup\{\langle at[\![\mathsf{S}]\!], \rho\rangle\langle at[\![\mathsf{S_t}]\!], \rho\rangle\pi | \mathcal{B}[\![\mathsf{B}]\!]\rho = True \wedge \langle at[\![\mathsf{S_t}]\!], \rho\rangle\pi \in \mathcal{S}[\![\mathsf{S_t}]\!]\}$$

اگر $\mathsf{Sl} ::= \epsilon$ باشد، ردهای پیشوندی متناظر با اجرای این عبارت دستوری را به شکل مجموعه‌ی زیر تعریف می کنیم:

$$\mathcal{S}[\![\mathsf{Sl}]\!] = \{\langle at[\![\mathsf{Sl}]\!], \rho\rangle | \rho \in \mathbb{EV}\}$$

اگر $\mathsf{Sl} ::= \mathsf{Sl'}\ \ \mathsf{S}$ باشد، ردهای پیشوندی متناظر با اجرای این عبارت دستوری را به شکل مجموعه‌ی زیر تعریف می کنیم:

$$\mathcal{S}[\![\mathsf{Sl}]\!] = \mathcal{S}[\![\mathsf{Sl'}]\!] \cup (\mathcal{S}[\![\mathsf{Sl'}]\!] \bowtie \mathcal{S}[\![\mathsf{S}]\!])$$

که اگر فرض کنیم $\mathcal{S}, \mathcal{S}'$ دو مجموعه شامل ردهای پیشوندی هستند، آنگاه عملگر چسباندن ($\bowtie$) روی آن‌ها به شکل زیر تعریف می‌شود:

$$\mathcal{S} \bowtie \mathcal{S}' = \{\pi \bowtie \pi' | \pi \in \mathcal{S} \wedge \pi' \in \mathcal{S}' \wedge (\pi \bowtie \pi'\ is\ well-defined)\}$$

اگر $S ::= while(B)S_b$ باشد، ماجرا نسبت به حالات قبل اندکی پیچیده‌تر می‌شود. تابعی به اسم $\mathcal{F}$ را تعریف خواهیم‌کرد که دو ورودی دارد. ورودی اول آن یک عبارت دستوری حلقه است و ورودی دوم آن یک مجموعه است. به عبارتی دیگر، به ازای هر حلقه یک تابع $\mathcal{F}$ جداگانه تعریف می‌شود که مجموعه‌ای از ردهای پیشوندی را می گیرد و مجموعه‌ای دیگر از همین موجودات را بازمی‌گرداند. کاری که این تابع انجام می‌دهد، این است که یک دور عبارت‌های دستوری داخل حلقه را اجرا می کند و دنباله‌هایی جدید را از دنباله‌های قبلی می‌سازد.

معنای یک حلقه را کوچکترین نقطه‌ثابت[22] این تابع در نظر می‌گیریم. در ادامه، تعریف $\mathcal{F}$ آمده است. با دیدن تعریف، می توان به دلیل این کار پی‌برد. ورودی‌ای که دیگر $\mathcal{F}$ روی آن اثر نکند، در دو حالت ممکن است اتفاق بیافتد. اولی این است که شرط حلقه برقرار نباشد. طبق تعریف $\mathcal{F}$، می‌توانیم ببینیم که $\mathcal{F}$ در این حالت چیزی به ردهای پیشوندی اضافه نمی‌کند.حالت دوم این است که اجرای برنامه داخل حلقه به عبارت دستوری break; برخورد کرده باشد که در این صورت وضعیتی به ته ردهای پیشوندی اضافه می‌شود که برچسبش خارج از مجموعه برچسب عبارت‌های دستوری حلقه است و همین اضافه کردن هر چیزی را به ته ردهای پیشوندی موجود، توسط $\mathcal{F}$ غیرممکن می‌کند.

بنابراین، نقطه‌ثابت[23] مفهوم مناسبی برای این است که از آن در تعریف صوری معنای حلقه استفاده شود. علت اینکه کوچکترین نقطه‌ثابت را به عنوان معنای حلقه در نظر می‌گیریم، این است که مطمئن هستیم، هر رد پیشوندی‌ای در نقطه‌ثابت موجود باشد، به گردش کار برنامه مرتبط است و ردهای پیشوندی اضافی و بی‌ربط به معنای برنامه، به آن وارد نمی‌شوند. برای درک بهتر این نکته می‌توان به این توجه کرد که با اضافه کردن وضعیت‌هایی کاملا بی‌ربط به گردش کار برنامه به ته ردهای پیشوندی، که صرفا برچسب متفاوتی با آخرین وضعیت هر رد پیشوندی دارند، نقطه‌ثابت جدیدی ساخته‌ایم. پس اگر خودمان را محدود به انتخاب کوچکترین نقطه‌ثابت نکنیم، به توصیفات صوری خوبی از برنامه‌ها دست پیدا نخواهیم‌کرد.

در مورد نقطه‌ثابت این نکته باقی می‌ماند که چه طور می‌توانیم مطمئن باشیم که چنین نقطه‌ثابتی وجود دارد. در این رابطه، باید گفت که مجموعه‌هایی که از ردهای پیشوندی تشکیل شده‌اند، با رابطه‌ی زیرمجموعه بودن یک را تشکیل می‌دهند. چون تابع S^* یکنوا است، بنا به قضیه‌ی نستر–تارسکی[24][۲۳] برای تابع S^* کوچکترین نقطه‌ثابت وجود دارد. تعاریف توابعی که در موردشان

[22] Least Fixpoint

[23] Fixpoint

[24] Knaster-Tarski Theorem

صحبت کردیم، به این شکل است:

$$\mathcal{S}[\![\mathsf{S}]\!] = lfp^{\subseteq}\, \mathcal{F}[\![\mathsf{S}]\!]$$

$$\mathcal{F}[\![\mathsf{S}]\!]X = \{\langle at[\![\mathsf{S}]\!], \rho\rangle | \rho \in \mathbb{EV}\} \cup$$

$$\{\pi_2\langle l, \rho\rangle\langle aft[\![\mathsf{S}]\!], \rho\rangle | \pi_2\langle l, \rho\rangle \in X \wedge \mathcal{B}[\![\mathsf{B}]\!]\rho = False \wedge l = at[\![\mathsf{S}]\!]\} \cup$$

$$\{\pi_2\langle l, \rho\rangle\langle at[\![\mathsf{S_b}]\!], \rho\rangle\pi_3 | \pi_2\langle l, \rho\rangle \in X \wedge \mathcal{B}[\![\mathsf{B}]\!]\rho = True \wedge$$

$$\langle at[\![\mathsf{S_b}]\!], \rho\rangle\pi_3 \in \mathcal{S}[\![\mathsf{S_b}]\!] \wedge l = at[\![\mathsf{S}]\!]\}$$

اگر ;=:: S باشد، ردهای پیشوندی متناظر با اجرای این عبارت دستوری را به شکل مجموعه‌ی زیر تعریف می کنیم:

$$\mathcal{S}[\![\mathsf{S}]\!] = \{\langle at[\![\mathsf{S}]\!], \rho\rangle | \rho \in \mathbb{EV}\} \cup \{\langle at[\![\mathsf{S}]\!], \rho\rangle\langle aft[\![\mathsf{S}]\!], \rho\rangle | \rho \in \mathbb{EV}\}$$

اگر {Sl} =:: S باشد، ردهای پیشوندی متناظر با اجرای این عبارت دستوری را به شکل مجموعه‌ی زیر تعریف می کنیم:

$$\mathcal{S}[\![\mathsf{S}]\!] = \mathcal{S}[\![\mathsf{Sl}]\!]$$

در اینجا، تعریف معنای برنامه‌ها به پایان می‌رسد.

فصل ۲

صوری‌گری جدید برای روش وارسی مدل

در این فصل، صورت جدیدی برای روش وارسی مدل ارائه می‌شود. در این صورت، برای بیان ویژگی‌های مورد بررسی به‌جای منطق زمانی از عبارات منظم استفاده می‌شود. با صحبت در مورد ویژگیهای برنامه‌ها شروع می‌کنیم، سپس به معرفی نحو و معناشناسی عبارات منظم، به عنوان یک منطق برای بیان ویژگی‌ها، می‌پردازیم و پس از آن، صورت جدید روش وارسی مدل را ارائه می‌کنیم. در آخر این فصل نیز، به بحث در مورد تصمیم پذیری این روش می‌پردازیم.

۱.۲ ویژگی‌های برنامه‌ها

تا به اینجای کار، نحو و معناشناسی یک زبان برنامه نویسی را تعریف کرده‌ایم. در این بخش، در مورد ویژگی‌های برنامه‌هایی که در این زبان نوشته می‌شوند، با توجه به معناشناسی رد پیشوندی که در فصل پیش تعریف کردیم، صحبت می‌کنیم. برای برنامه‌هایی که در یک زبان برنامه‌نویسی نوشته می‌شوند، می‌توان به اشکال مختلفی ویژگی تعریف کرد. مثلا ویژگی‌های کاملا نحوی، مثل اینکه طول برنامه چند خط است یا هر کاراکتر چند بار به کار رفته است، یا ویژگی‌های محاسباتی، مثل بررسی سرعت برنامه یا میزان استفاده‌ی آن از حافظه که عموما در نظریه الگوریتم و نظریه پیچیدگی محاسبه بررسی می‌شود. در اینجا، منظور ما از تعریف ویژگی، متناسب است با گردش کار برنامه‌ها. ابتدا به صحبت در مورد ویژگی‌ها در صوری‌گری‌مان، می‌پردازیم. صحبت در مورد اینکه با توجه به تعاریفی که ارائه داده‌ایم، خوب است که ویژگی‌ها را چه طور تعریف کنیم. سپس، به سراغ تعریف یک نوع عبارات منظم می‌رویم که بشود از آن برای بیان ویژگی‌ها استفاده کرد.

۱.۱.۲ ویژگی‌ها

همان‌طور که در فصل قبل دیدیم، معنای هر برنامه‌ی P با یک مجموعه‌ی $S^*[\![\mathsf{P}]\!]$ مشخص می‌شود. اگر بخواهیم، ویژگی‌هایی را برای موجوداتی ریاضیاتی که به با مجموعه‌ها تعریف شده‌اند، بیان بکنیم، اینکه ویژگی‌ها را هم با مجموعه‌ها بیان کنیم، کار معقولی به نظر می‌رسد. مثل اینکه بخواهیم ویژگی زوج بودن را در مورد اعداد طبیعی بیان کنیم. می توانیم مجموعه‌ی $\mathbb{E}$ را به عنوان مجموعه‌ی همه‌ی اعداد زوج در نظر بگیریم. در مورد یک گردایه از اعداد طبیعی، می‌توانیم خاصیت زوج بودن را معادل با این در نظر بگیریم که که هر چه داخل این گردایه هست، در $\mathbb{E}$ عضو است. پس یعنی در مورد گردایه‌هایی از اعداد طبیعی، می‌توان یک سری ویژگی را به شکل زیرمجموعه‌ای از مجموعه‌ی $\mathbb{N}$ بیان کرد. یعنی هر عضو $P(\mathbb{N})$ بنا به صحبت ما یک ویژگی از اعداد طبیعی است. در مورد برنامه‌ها نیز قرار است همین رویه را پیش بگیریم. تابع S^* از نوع $\mathbb{P} \rightarrow P(\mathfrak{S}^*)$ است. یعنی یک برنامه را در ورودی می‌گیرد و یک مجموعه از ردهای پیشوندی را باز می‌گرداند. پس می‌توانیم، هر ویژگی را به عنوان زیر مجموعه‌ای از $P(\mathfrak{S}^*)$ تعریف کنیم، به عبارت دیگر عضوی از $P(P(\mathfrak{S}^*))$. تعریف ویژگی‌ها به کمک رابطه‌ی زیرمجموعه بودن از سنت‌های نظریه‌ی تعبیر مجرد[۸] است.

۲.۱.۲ عبارات منظم

برای بیان ویژگی‌های برنامه‌ها به یک چارچوب نیاز داریم. در صورت سنتی روش وارسی مدل از منطق های زمانی برای بیان ویژگی‌ها به صورت صوری استفاده می‌کرد و این احتیاج به یک منطق را برای ارائه‌ی یک صوری‌گری کامل که رسیدن به صورت جدید وارسی مدل است، به ما نشان می‌دهد. طبق [۶] یک دلیل برای استفاده از عبارات منظم، که متداول‌تر بودن آن بین جامعه‌ی برنامه نویسان است. منظور این است که برنامه نویسان از شاخه‌های مختلف این تخصص همگی با عبارات منظم آشنا هستند، درحالیکه برنامه نویسانی که با منطق‌های زمانی آشنایی داشته باشند، بسیار کم هستند و جایگزینی اولی با دومی می‌تواند به رواج استفاده از روش وارسی مدل برای درستی‌یابی برنامه‌های کامپیوتری کمک کند. ابتدا، نحو این منطق را تعریف می‌کنیم، سپس به سراغ معناشناسی آن می‌رویم. عبارات منظمی که در اینجا تعریف می‌کنیم، با عبارات منظم سنتی که در علوم کامپیوتر رایج هستند، در جزئیات متفاوت هستند.

۳.۱.۲ نحو عبارات منظم

فرق عمده‌ای که نحو عبارات منظم ما با عبارات منظم سنتی دارد، در اتم[۱]هاست. اتم‌ها در عبارات منظم سنتی بدون ساختار بودند، اما در اینجا، ساختار دارند. در اینجا، هر اتم یک زوج مرتب است، متشکل از مجموعه‌ی L شامل برچسب‌ها و عبارت بولی B. این زوج مرتب را به شکل $\mathsf{L} : \mathsf{B}$ نمایش می‌دهیم.
نحو عبارات منظم به شکل زیر است:

تعریف ۱.۲.

$$\mathsf{L} \in P(\mathbb{L})$$

$$\mathsf{x}, \mathsf{y}, \ldots \in \mathbb{X}$$

$$\underline{\mathsf{x}}, \underline{\mathsf{y}}, \ldots \in \underline{\mathbb{X}}$$

$$\mathsf{B} \in \mathbb{B}$$

$$\mathsf{R} \in \mathbb{R}$$

$$\begin{aligned} \mathsf{R} ::= \quad & \varepsilon \\ \mid \quad & \mathsf{L} : \mathsf{B} \\ \mid \quad & \mathsf{R}_1\mathsf{R}_2 \quad (or\ \mathsf{R}_1 \bullet \mathsf{R}_2) \\ \mid \quad & \mathsf{R}_1 \mid \mathsf{R}_2 \quad (or\ \mathsf{R}_1 + \mathsf{R}_2) \\ \mid \quad & \mathsf{R}_1^* \\ \mid \quad & \mathsf{R}_1^+ \\ \mid \quad & (\mathsf{R}_1) \end{aligned}$$

همان طور که قابل مشاهده است، در اینجا، عملگرهای دوتایی چسباندن ($\bullet$) و انتخاب[۲] ($|$) را به همراه عملگرهای یگانی * و $^+$ داریم. در ادامه، با توجه به تعریف معناشناسی عبارات منظم[۳]،

[1] Atom
[2] Choice
[3] Regular Expressions Semantics

خواهیم دید که معنای عملگر یگانی $^{+}$ به وسیله‌ی عملگر یگانی * قابل بیان است. توجه شود که پرانتزها هم جزئی از نحو قرار داده شده‌اند.

همین طور در اینجا، می‌خواهیم از تعدادی عبارات مخفف که در ادامه استفاده می‌شوند، معرفی کنیم. منظور از عبارت منظم $\mathsf{B} : ?$ همان عبارت منظم $\mathsf{B} : \mathbb{L}$ است. عبارت منظم $l : \mathsf{B}$ به جای عبارت منظم $\{l\} : \mathsf{B}$ به کار می‌رود و منظور از عبارت منظم $\neg l : \mathsf{B}$ نیز عبارت منظم $\mathbb{L} \setminus \{l\} : \mathsf{B}$ است.

یک نکته‌ی قابل توجه، با توجه به تعاریف فصل قبل، وجود یک مجموعه به نام $\underline{\mathbb{X}}$ در کنار $\mathbb{X}$ که از قبل داشتیم، است. به ازای هر $\mathsf{x} \in \mathbb{X}$ یک $\underline{\mathsf{x}} \in \underline{\mathbb{X}}$ داریم. $\underline{\mathsf{x}}$ متغیر x در ابتدای هر برنامه است. همان‌طور که تعریف می‌کنیم:

تعریف ۲.۲. (محیط اولیه[4]): به هر تابع $\underline{\rho} : \underline{\mathbb{X}} \to \mathbb{V}$ یک محیط اولیه می‌گوییم. مجموعه‌ی همه‌ی محیط‌های اولیه را با $\underline{\mathbb{EV}}$ نمایش می‌دهیم.

۴.۱.۲ معناشناسی عبارات منظم

معنای هر عبارت منظم را با استفاده از تابع $\mathcal{S}^r$ به نام معناشناسی عبارات منظم[5] تعریف می‌کنیم. این تابع در ورودی‌اش یک عبارت منظم R را می‌گیرد، سپس یک مجموعه از زوج مرتب‌های $\langle \underline{\rho}, \pi \rangle$ را که $\pi \in \mathfrak{S}^*$ و $\underline{\rho} \in \underline{\mathbb{EV}}$ باز می‌گرداند. بنابراین این تابع از نوع $\mathbb{R} \to P(\underline{\mathbb{EV}} \times \mathfrak{S}^*)$ است. منظور از $\mathfrak{S}^*$ نیز $\mathfrak{S}^+ \cup \{\epsilon\}$ است.

تعریف استقرایی تابع $\mathcal{S}^r$ را در ادامه آورده‌ایم.

تعریف ۳.۲. (معناشناسی عبارات منظم): تابع $\mathcal{S}^r : \mathbb{R} \to P(\underline{\mathbb{EV}} \times \mathfrak{S}^*)$ به صورت استقرایی روی نحو عبارات منظم به صورت زیر تعریف می‌شود:

$$\mathcal{S}^r[\![\varepsilon]\!] = \{\langle \underline{\rho}, \epsilon \rangle | \underline{\rho} \in \underline{\mathbb{EV}}\}$$

[معنای عبارت منظم ε مجموعه‌ای شامل زوج مرتب‌هایی از محیط‌های اولیه‌ی مختلف در کنار رد پیشوندی تهی است.]

$$\mathcal{S}^r[\![\mathsf{L} : \mathsf{B}]\!] = \{\langle \underline{\rho}, \langle l, \rho \rangle \rangle | l \in \mathsf{L} \wedge \mathcal{B}[\![\mathsf{B}]\!]\underline{\rho}, \rho\}$$

[4] Initial Environment

[5] Semantics of Regular Expressions

[معنای عبارت منظم $\mathsf{L : B}$ مجموعه‌ای است شامل زوج مرتب‌هایی که عضو اول آن‌ها محیط‌های اولیه‌ی مختلف و عضو دوم آن‌ها ردهای پیشوندی تک‌عضوی $\langle l, \rho \rangle$ هستند. در این ردهای پیشوندی، برچسب l باید در L که مجموعه‌ای از برچسب‌هاست حضور داشته باشد. همین طور باید عبارت بولی B درباره‌ی محیط اولیه $\underline{\rho}$ و محیط ρ برقرار باشد. با توجه به حضور محیط‌های اولیه، در اینجا $\mathcal{B}$ به جای اینکه از نوع $\mathbb{EV} \rightarrow \mathbb{BOOL}$ باشد، از نوع $\underline{\mathbb{EV}} \rightarrow \mathbb{EV} \rightarrow \mathbb{BOOL}$ است(منظور از $\mathbb{BOOL}$ همان مجموعه‌ی $\{T, F\}$ است). بعد از تمام شدن تعریف معناشناسی عبارات منظم، دوباره $\mathcal{A}$ و $\mathcal{B}$ را با در نظر گرفتن محیط اولیه تعریف خواهیم کرد.]

$$\mathcal{S}^r[\![\mathsf{R_1 R_2}]\!] = \mathcal{S}^r[\![\mathsf{R_1}]\!] \bowtie \mathcal{S}^r[\![\mathsf{R_2}]\!]$$

به طوریکه، با فرض اینکه دو مجموعه‌ی $\mathcal{S}$ و $\mathcal{S}'$ هر یک معنای یک عبارت منظم باشند:

$$\mathcal{S} \bowtie \mathcal{S}' = \{\langle \underline{\rho}, \pi\pi' \rangle | \langle \underline{\rho}, \pi \rangle \in \mathcal{S} \wedge \langle \underline{\rho}, \pi' \rangle \in \mathcal{S}'\}$$

[اگر یک عبارت منظم داشته باشیم که از چسباندن دو عبارت منظم $\mathsf{R_1}$ و $\mathsf{R_2}$ به هم ساخته شده باشد، آنگاه معنای این عبارت منظم از زوج مرتب‌هایی تشکیل شده است که عضو اول آن‌ها محیط‌های اولیه هستند و عضو دوم آن‌ها از چسباندن ردهای پیشوندی موجود در عضو دوم اعضای مجموعه‌ی معنای این دو عبارت منظم تشکیل شده است. این عملگر چسباندن که در اینجا برای معنای عبارت‌های منظم تعریف شده است، با عملگر چسباندن برای معنای برنامه‌ها که در فصل پیش ارائه شد، متفاوت است. مورد دوم هم با نماد $\bowtie$ روی ردهای پیشوندی تعریف شد، اما در تعریفی که در این فصل از چسباندن ارائه داده‌ایم، این عملگر روی ردهای پیشوندی تعریف نشده است، بلکه روی زوج‌های مرتبی که در معنای هر عبارت منظمی حضور دارند، تعریف می‌شود.

تا این قسمت از تعریف معناشناسی عبارات منظم که رسیده‌ایم، تا حدی به درکی شهودی از اینکه به چه شکلی عبارات منظم راهی برای توصیف ویژگی در مورد برنامه‌ها هستند، دست پیدا کرده‌ایم. همان‌طور که در حالت قبل دیدیم، سازگاری هر زوج مرتب $\mathsf{L : B}$ با یک وضعیت داخل یک رد پیشوندی بررسی می‌شود. این زوج مرتب‌ها موازی با وضعیت‌ها در ردهای پیشوندی معنای برنامه پیش می‌روند و سازگاری آن‌ها با یکدیگر بررسی می‌شود تا وارسی مدل انجام شود.]

$$\mathcal{S}^r[\![\mathsf{R_1} \mid \mathsf{R_2}]\!] = \mathcal{S}^r[\![R_1]\!] \cup \mathcal{S}^r[\![R_2]\!]$$

[این حالت، معنای اعمال عملگر انتخاب روی دو عبارت منظم را توصیف می‌کند. معنای اعمال این عملگر به صورت اجتماع معنای هر دو عبارت منظم تعریف شده است.]

$$\mathcal{S}^r[\![\mathsf{R}]\!]^0 = \mathcal{S}^r[\![\varepsilon]\!]$$

$$\mathcal{S}^r[\![\mathsf{R}]\!]^{n+1} = \mathcal{S}^r[\![\mathsf{R}]\!]^n \bowtie \mathcal{S}^r[\![\mathsf{R}]\!]$$

[دو عبارت اخیر، برای تعریف معنای عملگرهای * و $^+$ تعریف شده‌اند. عملگر $\bowtie$ و معنای $\mathcal{S}^r[\![\varepsilon]\!]$ را هم که قبلا تعریف کرده بودیم و 0 و n هم اعداد طبیعی‌اند.]

$$\mathcal{S}^r[\![\mathsf{R}^*]\!] = \bigcup_{n\in\mathbb{N}} \mathcal{S}^r[\![\mathsf{R}^n]\!]$$

$$\mathcal{S}^r[\![\mathsf{R}^+]\!] = \bigcup_{n\in\mathbb{N}\setminus\{0\}} \mathcal{S}^r[\![\mathsf{R}^n]\!]$$

[این دو عبارت تعریف معنای دو عملگر * و $^+$ هستند. منظور از $\mathbb{N}$ مجموعه‌ی اعداد طبیعی است. همان‌طور که قبل‌تر هم اشاره شد، معنای عملگر $^+$ با عملگر * قابل بیان است. اضافه می‌کنیم که معنای عملگر * را در فرازبان(و نه در منطق تعریف شده در اینجا) می‌توان با عملگر انتخاب تعریف کرد.]

$$\mathcal{S}^r[\![(\mathsf{B})]\!] = \mathcal{S}^r[\![\mathsf{B}]\!]$$

[این قسمت از تعریف هم صرفا بیان می‌کند که پرانتزها تاثیری در معنای عبارات منظم ندارند که کاملا قابل انتظار است، چرا که وجود پرانتز قرار است صرفا در خواص نحو منطق نقش داشته باشد.]

تعریف ۴.۲. (ارضا کردن): می‌گوییم، در محیط اولیه‌ی $\underline{\rho}$ رد پیشوندی π عبارت منظم R را ارضا می‌کند، اگر و تنها اگر $\langle\underline{\rho}, \pi\rangle \in \mathcal{S}^r[\![\mathsf{R}]\!]$.

تعریف معناشناسی عبارات منظم در اینجا تمام می‌شود، اما همان‌گونه که در لابه‌لای تعاریف گفته شد، احتیاج داریم که $\mathcal{A}$ و $\mathcal{B}$ را از نو تعریف کنیم:

تعریف ۵.۲. توابع $\mathcal{A} : \mathbb{A} \to \underline{\mathbb{EV}} \to \mathbb{EV} \to \mathbb{V}$ و $\mathcal{B} : \mathbb{B} \to \underline{\mathbb{EV}} \to \mathbb{EV} \to \mathbb{BOOL}$ به شکل استقرایی به ترتیب روی ساختار عبارت‌های حسابی $\mathsf{A} \in \mathbb{A}$ و عبارت‌های بولی $\mathsf{B} \in \mathbb{B}$ به شکل زیر تعریف می‌شوند:

$$\mathcal{A}[\![1]\!]\underline{\rho}, \rho = 1$$

$$\mathcal{A}[\![\underline{\mathsf{x}}]\!]\underline{\rho}, \rho = \underline{\rho}(\mathsf{x})$$

$$\mathcal{A}[\![\mathsf{x}]\!]\underline{\rho}, \rho = \rho(\mathsf{x})$$

$$\mathcal{A}[\![\mathsf{A}_1 - \mathsf{A}_2]\!]\underline{\rho}, \rho = \mathcal{A}[\![\mathsf{A}_1]\!]\underline{\rho}, \rho - \mathcal{A}[\![\mathsf{A}_2]\!]\underline{\rho}, \rho$$

$$\mathcal{B}[\![\mathsf{A}_1 < \mathsf{A}_2]\!]\underline{\rho}, \rho = \mathcal{A}[\![\mathsf{A}_1]\!]\underline{\rho}, \rho \ < \ \mathcal{A}[\![\mathsf{A}_2]\!]\underline{\rho}, \rho$$

$$\mathcal{B}[\![\mathsf{B}_1 \text{ nand } \mathsf{B}_2]\!]\underline{\rho}, \rho = \mathcal{B}[\![\mathsf{B}_1]\!]\underline{\rho}, \rho \uparrow \mathcal{B}[\![\mathsf{B}_2]\!]\underline{\rho}, \rho$$

قابل مشاهده است که تعاریف جدید تا حد خوبی به تعاریف قبلی شبیه هستند و فرق عمده صرفا وارد شدن $\underline{\rho}$ است.

حال که معناشناسی عبارات منظم را داریم، به طور مختصر به مقایسه‌ی عبارات منظمی که در این بحث تعریف کرده‌ایم و عبارات منظم کلاسیک در باقی نوشته‌ها و موضوعات می‌پردازیم. جبر کلاینی[۶] یک ساختار جبری است که تعمیمی است از عبارات منظم معرفی شده در [۱۵]. سر و کله‌ی عبارات منظم در قسمت‌های مختلفی از علوم کامپیوتر پیدا می‌شود، اما با تعاریفی که هم‌ارز نیستند. هدف از ارائه‌ی جبر کلاینی این بوده است که تعریفی عمومی باشد که این تعاریف غیر هم‌ارز را در خود جای می‌دهد. در [۱۶] آمده است که برای جبر کلاینی هم تعاریف مختلفی که با هم‌ارز نیستند، معرفی شده است. علاوه بر این، این مقاله به بررسی این تعاریف و ارتباطشان با یکدیگر پرداخته است. همین طور، این مقاله خود با یک تعریف از جبر کلاینی شروع کرده است. طبق این تعریف، اگر عبارات منظمی که در اینجا تعریف کرده‌ایم، یک جبر کلاینی می‌بودند، می‌بایستی که برای هر عبارت منظم R می‌داشتیم $\mathcal{S}^r[\![\varepsilon\mathsf{R}]\!] = \mathcal{S}^r[\![\varepsilon]\!]$، چون ε نقش صفر عملگر چسباندن(که عملگر ضرب جبر کلاینی است) را دارد و یک جبر کلاینی برای صفر قانون جذب را به عنوان یک اصل دارد. اما در مورد عبارات منظمی که در اینجا تعریف کرده‌ایم، داریم $\mathcal{S}^r[\![\varepsilon\mathsf{R}]\!] = \mathcal{S}^r[\![\mathsf{R}]\!]$. بیشتر از این به این بحث نمی‌پردازیم که بحث دامنه‌دار و منحرف کننده‌ایست. یک قضیه را در مورد عبارات منظم ارائه می‌دهیم که پخش پذیری عملگر انتخاب به عملگر چسباندن است و بعد به ادامه‌ی راه اصلیمان می‌پردازیم.

قضیه ۶.۲. برای عبارت‌های منظم $\mathsf{R}, \mathsf{R}_1, \mathsf{R}_2$ داریم:

$$\mathcal{S}^r[\![\mathsf{R} \bullet (\mathsf{R}_1 + \mathsf{R}_2)]\!] = \mathcal{S}^r[\![(\mathsf{R} \bullet \mathsf{R}_1) + (\mathsf{R} \bullet \mathsf{R}_2)]\!]$$

[6] Kleene Algebra

اثبات.

$$\mathcal{S}^r[\![\mathsf{R} \bullet (\mathsf{R}_1 + \mathsf{R}_2)]\!] = \mathcal{S}^r[\![\mathsf{R}]\!] \bowtie \mathcal{S}^r[\![(\mathsf{R}_1 + \mathsf{R}_2)]\!]$$

$$= \mathcal{S}^r[\![\mathsf{R}]\!] \bowtie \mathcal{S}^r[\![\mathsf{R}_1 + \mathsf{R}_2]\!] = \mathcal{S}^r[\![\mathsf{R}]\!] \bowtie (\mathcal{S}^r[\![\mathsf{R}_1]\!] \cup \mathcal{S}^r[\![\mathsf{R}_2]\!])$$

$$= \{\langle \underline{\rho}, \pi\pi' \rangle | \langle \underline{\rho}, \pi \rangle \in \mathcal{S}^r[\![\mathsf{R}]\!] \wedge (\langle \underline{\rho}, \pi' \rangle \in \mathcal{S}^r[\![\mathsf{R}_1]\!] \vee \langle \underline{\rho}, \pi' \rangle \in \mathcal{S}^r[\![\mathsf{R}_2]\!])\}$$

$$= \{\langle \underline{\rho}, \pi\pi' \rangle | (\langle \underline{\rho}, \pi \rangle \in \mathcal{S}^r[\![\mathsf{R}]\!] \wedge \langle \underline{\rho}, \pi' \rangle \in \mathcal{S}^r[\![\mathsf{R}_1]\!]) \vee (\langle \underline{\rho}, \pi \rangle \in \mathcal{S}^r[\![\mathsf{R}]\!] \wedge \langle \underline{\rho}, \pi' \rangle \in \mathcal{S}^r[\![\mathsf{R}_2]\!]\}$$

$$= \{\langle \underline{\rho}, \pi\pi' \rangle | \langle \underline{\rho}, \pi \rangle \in \mathcal{S}^r[\![\mathsf{R}]\!] \wedge \langle \underline{\rho}, \pi' \rangle \in \mathcal{S}^r[\![\mathsf{R}_1]\!]\} \cup \{\langle \underline{\rho}, \pi\pi' \rangle | \langle \underline{\rho}, \pi \rangle \in \mathcal{S}^r[\![\mathsf{R}]\!] \wedge \langle \underline{\rho}, \pi' \rangle \in \mathcal{S}^r[\![\mathsf{R}_2]\!]\}$$

$$= (\mathcal{S}^r[\![\mathsf{R}]\!] \bowtie \mathcal{S}^r[\![\mathsf{R}_1]\!]) \cup (\mathcal{S}^r[\![\mathsf{R}]\!] \bowtie \mathcal{S}^r[\![\mathsf{R}_2]\!]) = \mathcal{S}^r[\![\mathsf{R} \bullet \mathsf{R}_1]\!] \cup \mathcal{S}^r[\![\mathsf{R} \bullet \mathsf{R}_2]\!]$$

$$= \mathcal{S}^r[\![(\mathsf{R} \bullet \mathsf{R}_1) + (\mathsf{R} \bullet \mathsf{R}_2)]\!]$$

$\square$

تا اینجای کار، بیشتر مفاهیمی که برای بیان صورت جدید روش وارسی مدل احتیاج داریم را بیان کرده‌ایم، هرچند که هنوز برخی از تعاریف باقی مانده‌اند.

۵.۱.۲ گونه‌های مختلف نحو عبارات منظم

به عنوان قسمت آخر این بخش، گونه‌های مختلفی از نحو عبارات منظم را بیان می‌کنیم که هر کدام در واقع زیرمجموعه‌ای از منطقی که توصیف کرده‌ایم را تشکیل می‌دهند. بعضی از آن‌ها را در همین فصل، برای هدف نهایی این فصل و بعضی دیگر را در فصل بعدی استفاده می‌کنیم.

می‌توانیم به هر یک از این عبارات منظم به چشم یک منطق متفاوت نگاه کنیم. هرچند که معناشناسی عبارات منظم به عنوان معناشناسی هر یک از این منطق‌ها باقی می‌ماند. با توجه به اینکه هر یک از این منطق‌ها زیرمجموعه‌ی عبارات منظم هستند، می‌توان گفت معناشناسی هر یک از آن‌ها همان تابع $\mathcal{S}^r$ با دامنه‌ی محدود شده است.

اولین گونه‌ای که می‌خواهیم بیان کنیم، گونه‌ای است که در اعضای آن اصلا اتم حضور ندارد و کل عبارت‌های منظم عضو این منطق از εها تشکیل شده‌اند.

تعریف ۷.۲. (عبارات منظم تهی[7] ـ $\mathbb{R}_\varepsilon$):

$$\mathsf{R} \in \mathbb{R}_\varepsilon$$

[7] Empty Regular Expressions

$$\mathsf{R} \quad ::= \quad \varepsilon \mid \mathsf{R}_1\mathsf{R}_2 \mid \mathsf{R}_1 + \mathsf{R}_2 \mid \mathsf{R}_1^* \mid \mathsf{R}_1^+ \mid (\mathsf{R}_1)$$

با توجه به آنچه گفتیم، معنای همه‌ی این عبارت‌های منظم عضو این منطق برابر $\{\langle \underline{\rho}, \epsilon \rangle\}$ خواهد بود.

گونه‌ی بعدی عبارات منظم ناتهی[۸] است.

تعریف ۲.۸. (عبارات منظم ناتهی ـ $\mathbb{R}^+$):

$$\mathsf{R} \quad \in \quad \mathbb{R}^+$$

$$\mathsf{R} \quad ::= \quad \mathsf{L} : \mathsf{B} \mid \varepsilon\mathsf{R}_2 \mid \mathsf{R}_1\varepsilon \mid \mathsf{R}_1\mathsf{R}_2 \mid \mathsf{R}_1 + \mathsf{R}_2 \mid \mathsf{R}_1^+ \mid (\mathsf{R}_1)$$

دلیل وجود $\varepsilon\mathsf{R}_2$ و $\mathsf{R}_1\varepsilon$ در تعریف این است که ممکن است معنای عبارت منظمی در این منطق با $\langle \underline{\rho}, \epsilon \rangle$ برابر نباشد، اما در آن عبات منظم، ε حضور داشته باشد. با این تفاصیل، می‌توان دید که دو مجموعه‌ی $\mathbb{R}_\varepsilon$ و $\mathbb{R}^+$ یک افراز برای مجموعه‌ی $\mathbb{R}$ هستند، چونکه معنای هر عبارت منظم در $\mathbb{R}$ یا با $\langle \underline{\rho}, \epsilon \rangle$ برابر هست یا نیست. بنابراین شاید به نظر برسد که تعریف یکی از این نحوها کافی باشد. اما این‌طور نیست، چون ممکن است که درجایی احتیاج داشته باشیم که تعریفی استقرایی روی هر یک از این دو ساختار ارائه دهیم، یا اینکه در اثبات حکمی بخواهیم از استقرا روی یکی از این دو ساختار استفاده کنیم.

گونه‌ی آخر عبارات منظم ما نیز عبارات منظم بدون انتخاب[۹] است.

تعریف ۲.۹. (عبارات منظم بدون انتخاب ـ $\mathbb{R}^\dagger$):

$$\mathsf{R} \quad \in \quad \mathbb{R}^\dagger$$

$$\mathsf{R} \quad ::= \quad \varepsilon \mid \mathsf{L} : \mathsf{B} \mid \mathsf{R}_1\mathsf{R}_2 \mid {\mathsf{R}_1}^* \mid {\mathsf{R}_1}^+ \mid (\mathsf{R}_1)$$

۲.۲ صورت جدید مسئله‌ی وارسی مدل

بالاخره، به هدف نهایی این فصل رسیدیم. می‌خواهیم صورت جدیدی از مسئله‌ی وارسی مدل را بیان کنیم.

[8] Non-empty Regular Expressions
[9] Choice-free Regular Expressions

پیش از ارائه‌ی تعریف وارسی مدل، نیاز داریم که عملگر بستار پیشوندی[10] را برای یک مجموعه از ردهای پیشوندی معرفی کنیم.

تعریف ۱۰.۲. (بستار پیشوندی): اگر $\Pi \in P(\underline{\mathbb{EV}} \times \mathfrak{S}^+)$، آنگاه بستار پیشوندی Π را به صورت زیر تعریف می‌کنیم:

$$\mathsf{prefix}(\Pi) = \{\langle \underline{\rho}, \pi\rangle | \pi \in \mathfrak{S}^+ \land \exists\, \pi' \in \mathfrak{S}^* : \langle \underline{\rho}, \pi\pi'\rangle \in \Pi\}$$

برای درک بهتر مفهوم بستار پیشوندی به مثال زیر توجه شود.

مثال ۱۱.۲. اگر $\Pi = \{\langle \underline{\rho}, \langle l_1, \rho_1\rangle\langle l_2\rho_2\rangle\rangle\langle l_3, \rho_3\rangle\rangle, \langle \underline{\rho}, \langle {l_1}', {\rho_1}'\rangle\langle {l_2}'{\rho_2}'\rangle\rangle\}$ باشد(Π شامل دو عضو است)، آنگاه:

$$\mathsf{prefix}(\Pi) = \{\langle \underline{\rho}, \langle l_1, \rho_1\rangle\rangle, \langle \underline{\rho}, \langle l_1, \rho_1\rangle\langle l_2\rho_2\rangle\rangle, \langle \underline{\rho}, \langle l_1, \rho_1\rangle\langle l_2\rho_2\rangle\langle l_3, \rho_3\rangle\rangle,$$
$$\langle \underline{\rho}, \langle {l_1}', {\rho_1}'\rangle\rangle, \langle \underline{\rho}, \langle {l_1}', {\rho_1}'\rangle\langle {l_2}'{\rho_2}'\rangle\rangle\}$$

که شامل ۵ عضو است.

حال به ارائه‌ی صورت جدید روش وارسی مدل می‌رسیم.

تعریف ۱۲.۲. (وارسی مدل): اگر $\mathsf{P} \in \mathbb{P}, \mathsf{R} \in \mathbb{R}^+, \underline{\rho} \in \underline{\mathbb{EV}}$ آنگاه:

$$\mathsf{P}, \underline{\rho} \models \mathsf{R} \Leftrightarrow (\{\underline{\rho}\} \times \mathcal{S}^*[\![\mathsf{P}]\!]) \subseteq \mathsf{prefix}(\mathcal{S}^r[\![\mathsf{R} \bullet (?:T)^*]\!])$$

این تعریف بیان می‌کند که اگر برنامه‌ی P با محیط اولیه‌ی $\underline{\rho}$ اجرا شود، این برنامه در صورتی ویژگی‌ای که با یک عبارت منظم R بیان شده است را دارد که معنای آن زیرمجموعه‌ی بستار پیشوندی معنای عبارت منظم $\mathsf{R} \bullet (?:T)^*$ باشد. توجه شود که محیط اولیه‌ای که برای برنامه‌ی مورد بررسی در نظر گرفته‌ایم، صرفا به این منظور در تعریف قرار داده شده است که بتوانیم، معنای برنامه‌ها را با معنای عبارت‌های منظم قابل قیاس کنیم. دلیل حضور محیط اولیه در معنای عبارت‌های منظم نیز در صورت سوم روش وارسی مدل مشخص می‌شود، یعنی جایی که وارسی مدل روی ساختار برنامه‌ها تعریف شده است و ردهای پیشوندی موجود در معنای هر عبارت دستوری یا لیست عبارت‌های دستوری با محیط متفاوتی شروع می‌شوند و اطلاعات محیط اولیه‌ی برنامه در این ردها حضور ندارد

[10] Prefix Closure

(با اینکه ممکن است به آن نیاز داشته باشیم). در این صورت از روش وارسی مدل و صورت بعدی، محیط‌های اولیه صرفا حضور دارند و در عمل نقش مهمی ندارند.

در مورد نقش عبارت منظم $(? : T)^*$ و عملگر prefix نیز می‌توان گفت، بنا به تصمیم مبدع این روش، اگر یک رد پیشوندی، با عبارت منظم ناسازگاری نداشته باشد و از آن کوتاه‌تر باشد، این رد پیشوندی با ویژگی بیان شده را دارد. این نکته صرفا در مورد حضور prefix بود. حضور $(? : T)^*$ نیز باعث می‌شود اگر طول عبارت منظم مورد بررسی کمتر از رد پیشوندی باشد و ناسازگاری‌ای مشاهده نشده باشد، رد پیشوندی دارای ویژگی R در نظر گرفته شود. در اینجا، بیان صورت جدید وارسی مدل به اتمام می‌رسد.

۳.۲ در مورد توقف پذیری

در این بخش نکته‌ای در مورد کار که به نظر نگارنده رسیده، مطرح شده است. اگر صحبت ما در اینجا درست باشد، این به این معنی خواهد بود که روشی که در حال توصیفش هستیم،تصمیم ناپذیر است، به عبارت دیگر قابلیت پیاده‌سازی را ندارد!

بحث ما در اینجا در مورد توقف پذیری است. در [۶] در مورد تعریف توقف یک برنامه صحبتی به میان نیامده است، یعنی گفته نشده است که در چه صورتی می‌توانیم، بگوییم که یک برنامه متوقف شده است. یک تعریف صوری که خودمان می‌توانیم برای این مفهوم، بیاوریم این است:

تعریف ۱۳.۲. (توقف پذیری:) برنامه‌ی P را به همراه محیط اولیه $\underline{\rho}$ توقف پذیر[11] می‌گوییم، اگر و تنها اگر وجود داشته باشد $\pi \in \mathcal{S}^*[\![\mathsf{P}]\!]$، که ($\rho$ محیط متناظر با محیط اولیه‌ی $\underline{\rho}$ است):

$$\pi = \langle at[\![\mathsf{P}]\!], \rho \rangle \pi'$$

و اینکه $\langle aft[\![\mathsf{P}]\!], \rho' \rangle$ در π حضور داشته باشد. این اتفاق را با $\mathsf{P}, \underline{\rho} \downarrow$ نشان می‌دهیم. همین تعریف را برای لیست عبارت‌های دستوری SI یا عبارت دستوری S صرفا با جایگذاری به‌جای برنامه‌ی P نیز داریم.

در این تعریف، توقف پذیری صرفا برای یک محیط اولیه تعریف شده است. در اینجا، توقف پذیری به متناهی بودن ردهای پیشوندی موجود در برنامه ارتباطی ندارد. با توجه به معناشناسی رد

[11] Terminable

پیشوندی، تعریف کردن توقف پذیری به معنای وجود رد پیشوندی متناهی در معنای برنامه به هیچ عنوان مناسب نیست، چون معناشناسی خاصیت پیشوندی بودن را دارد و مطمئن هستیم، در معنای هر برنامه‌ای حتما یک رد پیشوندی متناهی با محیط اولیه‌ی مورد بررسی وجود دارد.

اگر هم بخواهیم، تعریف توقف پذیری را عدم حضور ردهای پیشوندی نامتناهی در معنای برنامه در نظر بگیریم، به تعریف قوی‌تری نسبت به آنچه ارائه دادیم، رسیده‌ایم. در اینجا، ما سعی می‌کنیم، ضعیف‌ترین تعریف معقول را ارائه دهیم. در این صورت، از اثباتی که بر اساس تعریف ارائه شده، برای تصمیم ناپذیری می‌آوریم، مطمئن‌تر خواهیم شد. با این حال، در قضیه‌ی بعدی می‌بینیم که تعریفی که ارائه کرده‌ایم با اینکه بگوییم، در معنای برنامه رد پیشوندی نامتناهی وجود ندارد معادل است.

قضیه ۱۴.۲ . برای برنامه‌ی P و محیط اولیه‌ی $\underline{\rho}$ داریم $\mathsf{P}, \underline{\rho} \downarrow$، اگر و تنها اگر ρ محیط متناظر با محیط اولیه‌ی $\underline{\rho}$ باشد و

$$\forall \pi \in \mathfrak{S}^{+} : \langle at[\![\mathsf{P}]\!], \rho\rangle\pi \in \mathcal{S}^{*}[\![\mathsf{P}]\!] \rightarrow \langle at[\![\mathsf{P}]\!], \rho\rangle\pi \in \mathbb{R}^{+}$$

اثبات. $(\Rightarrow)$ برای این حالت، باید ثابت کنیم که در معنای هر برنامه‌ای رد پیشوندی‌ای وجود دارد که با $\langle at[\![\mathsf{P}]\!], \rho\rangle$ شروع شده است و به ازای یک محیط ρ' به $\langle aft[\![\mathsf{P}]\!], \rho'\rangle$ ختم شده است. در این اثبات، از تعریف برچسب‌ها که در ضمیمه‌ی [۶] آمده است، استفاده کرده‌ایم. داریم: $\mathsf{P} = \mathsf{Sl}$ و $aft[\![\mathsf{P}]\!] = aft[\![\mathsf{Sl}]\!]$.

حکم را با استقرا روی ساختار Sl ثابت می‌کنیم.

$$\blacktriangleright\ \mathsf{Sl} = \ni :$$

داریم:

$$\mathcal{S}^{*}[\![\ni]\!] = \{\langle at[\![\ni]\!], \dot{\rho}\rangle | \dot{\rho} \in \mathbb{EV}\}$$

همین طور طبق تعریف برچسب‌ها داریم:

$$at[\![\ni]\!] = aft[\![\ni]\!]$$

پس حکم برقرار است.

$$\blacktriangleright\ \mathsf{Sl} = \mathsf{Sl}'\ \mathsf{S} :$$

اینکه در معنای SI، رد پیشوندی‌ای شامل $\langle aft[\![\mathsf{SI}]\!], \rho'\rangle$ وجود داشته باشد، با توجه به تعاریفی که داریم، به این وابسته است که در معنای S، دنباله‌ای شامل $\langle aft[\![\mathsf{S}]\!], \rho'\rangle$ وجود داشته باشد. برای اینکه این گزاره را ثابت کنیم، باید آن را روی ساختار S ثابت کنیم که در واقع بخش اصلی اثبات این سمت قضیه است.

$$\blacktriangleright\blacktriangleright\ \mathsf{S}\ =\ \mathsf{x} \doteq \mathsf{A};:$$

در این حالت، با توجه به تعریف معنای S، دنباله‌ی

$$\langle at[\![\mathsf{S}]\!], \rho\rangle\langle aft[\![\mathsf{S}]\!], \rho[\mathsf{x} \leftarrow \mathcal{A}[\![\mathsf{A}]\!]\rho]\rangle$$

در معنای عبارت دستوری به ازای هر ρ وجود دارد. پس دنباله‌ای به شکل بالا با محیطی متناظر با $\underline{\rho}$ هم در معنای عبارت دستوری حضور دارد.

$$\blacktriangleright\blacktriangleright\ \mathsf{S} = ;\quad :$$

با توجه به معنای این عبارت دستوری، دنباله‌ی زیر در معنای این عبارت دستوری وجود دارد.

$$\langle at[\![\mathsf{S}]\!], \rho\rangle\langle aft[\![\mathsf{S}]\!], \rho\rangle$$

$$\blacktriangleright\blacktriangleright\ \mathsf{S} =\ \mathsf{if}\ (\mathsf{B})\ \mathsf{S_t}:$$

در صورتی که $\mathcal{B}[\![\mathsf{B}]\!]\rho = T$ برقرار باشد، دنباله‌ی

$$\langle at[\![\mathsf{S}]\!], \rho\rangle\langle at[\![\mathsf{S_t}]\!], \rho\rangle\pi$$

در مجموعه‌ی معنای این عبارت دستوری حضور دارد، در حالیکه، $\langle at[\![\mathsf{S_t}]\!], \rho\rangle\pi$ داخل معنای $\mathsf{S_t}$ نیز هست. طبق فرض استقرا می‌دانیم که برچسب آخرین وضعیت π برابر است با $aft[\![\mathsf{S_t}]\!]$ که طبق تعاریف مربوط به برچسب‌ها داریم $aft[\![\mathsf{S_t}]\!] = aft[\![\mathsf{S}]\!]$.
در صورتی که معنای عبارت بولی غلط باشد نیز، طبق تعریف، دنباله‌ی زیر در معنای عبارت دستوری موجود است.

$$\langle at[\![\mathsf{S}]\!], \rho\rangle\langle aft[\![\mathsf{S}]\!], \rho\rangle$$

$$\blacktriangleright\blacktriangleright\ \mathsf{S} =\ \mathsf{if}\ (\mathsf{B})\ \mathsf{S_t}\ \mathsf{else}\ \mathsf{S_f}:$$

مانند حالت قبل است، منتها با این تفاوت که در صورتی که معنای عبارت بولی غلط باشد، دنباله‌ی زیر در معنای عبارت دستوری حضور دارد:

$$\langle at[\![\mathsf{S}]\!], \rho\rangle\langle at[\![\mathsf{S_f}]\!], \rho\rangle\pi$$

همین طور، تساوی $aft[\![\mathsf{S_t}]\!] = aft[\![\mathsf{S}]\!] = aft[\![\mathsf{S_f}]\!]$ هم طبق تعریف برچسب‌ها برقرار است.

$$\blacktriangleright\blacktriangleright\ \mathsf{S} = \ \mathsf{while}\ (\mathsf{B})\ \mathsf{S_t}\ :$$

در اثبات این سمت قضیه، این حالت پیچیده ترین حالت است و در واقع تنها حالتی است که در اثبات آن به فرض قضیه احتیاج داریم! همان طور که پیش‌تر گفتیم، معنای عبارت دستوری حلقه به کمک یک تابع تعریف می‌شود. معنای حلقه کوچکترین نقطه‌ثابت این تابع است، در حالیکه، این تابع وقتی روی یک مجموعه از ردهای پیشوندی اعمال شود، تاثیرات یک بار اجرای عبارت(های) دستوری درون حلقه را روی ردهای پیشوندی درون مجموعه اعمال می‌کند.

طبق تعریف $\mathcal{F}$، مطمئن هستیم که رد پیشوندی‌ای که با محیط $\underline{\rho}$ شروع شود، در مجموعه‌ی معنای S وجود دارد، چون به ازای هر محیط $\acute{\rho}$(نقطه به این خاطر است که با ρ خاص موجود در فرض اشتباه گرفته نشود) حالت $\langle at[\![\mathsf{S}]\!], \acute{\rho}\rangle$ در هر اعمال تابع $\mathcal{F}$ روی هر مجموعه‌ی دلخواه وجود دارد. چون معنای S را به عنوان کوچک‌ترین نقطه‌ثابت $\mathcal{F}$ در نظر گرفته‌ایم، مطمئن هستیم که مجموعه‌ای که کوچکترین نقطه‌ثابت تابع $\mathcal{F}$ است، شامل رد پیشوندی $\langle at[\![\mathsf{S}]\!], \rho\rangle$ می‌شود. این رد پیشوندی، با اجرای $\mathcal{F}$ تحت تاثیر قرار می‌گیرد. اگر معنای B در یکی از اعمال های $\mathcal{F}$ غلط باشد، رد پیشوندی $\langle at[\![\mathsf{S}]\!], \rho\rangle\pi\langle aft[\![\mathsf{S}]\!], \rho'\rangle$ در معنای برنامه قرار خواهد گرفت و می‌توانیم، بگوییم اجرای عبارت دستوری با این محیط اولیه توقف پذیر است. می‌دانیم که طبق تعریف تابع، چیزی به انتهای این رد پیشوندی اضافه نمی‌شود. از طرف دیگر، با این محیط اولیه، با توجه به تعریف، رد پیشوندی دیگری وجود ندارد که طولانی‌تر از رد پیشوندی مورد اشاره باشد.

در حالت دیگر، اگر فرض کنیم هیچ گاه به حالتی نمی‌رسیم که در آن معنای B غلط باشد هم با فرض مسئله به تناقض می‌خوریم، چون در آن صورت تابع $\mathcal{F}$ مدام به دنباله‌هایی که با محیط ρ شروع می‌شوند، می‌افزاید و این یک دنباله‌ی نامتناهی را خواهد ساخت. در صورتی که معنای B هیچ گاه صحیح نباشد، حداقل حالت $\langle at[\![\mathsf{S_t}]\!], \rho''\rangle$ به آخر دنباله‌های پیشین اضافه خواهد شد و از این جهت مطمئن هستیم که دنباله‌ی نامتناهی گفته شده در معنای عبارت دستوری حضور خواهد داشت.

پس با این تفاصیل، این حالت هم ثابت می‌شود.

$\blacktriangleright\blacktriangleright$ S = break; :

در تعریف تابع aft روی برچسب‌ها در [۶] تعریف برای این عبارت دستوری مشخص نیست! در [۷] هم در مورد این برچسب‌ها بحث شده است. در آنجا، گفته شده است که در مورد بخش هایی از تعریف این توابع، که تعریف مشخصی ارائه نشده است، برداشت آزاد است! ما در اینجا سعی داریم، معقول ترین برداشتی که نسبت به درکمان از این کار می‌توانیم داشته باشیم را بیان کنیم. مهم‌ترین چیزی که در مورد برچسب‌ها در این عبارت دستوری باید برقرار باشد، این است که اگر این عبارت دستوری بخشی از $\mathsf{S_t}$ در حلقه‌ی زیر باشد،

$$\mathsf{S'} = \text{ while } (\mathsf{B})\ \mathsf{S_t}$$

در این صورت، تساوی $aft[\![\mathsf{S'}]\!] = brk{-}to[\![\mathsf{S_t}]\!]$ را طبق تعریف خواهیم داشت. انتظار می‌رود که $aft[\![\text{break;}]\!] = aft[\![\mathsf{S'}]\!]$ باشد. اینکه اجرای برنامه پس از اجرای ;break بعد از عبارت دستوری $\mathsf{S'}$ پی گرفته شود، انتظار معقولی است از صوری‌گری‌ای که در حال توصیف گردش کار کامپیوتری است. البته در نظر گرفته شود که فرض کرده‌ایم که $\mathsf{S'}$ داخلی ترین حلقه‌ای است که ;break درون آن جای دارد.

از پس این فرض‌های ما $aft[\![\text{break;}]\!] = break - to[\![\mathsf{S_t}]\!]$ نتیجه می‌شود و طبق معنای عبارت دستوری ;break، رد پیشوندی زیر در معنای این عبارت دستوری وجود دارد،

$$\langle at[\![\text{break;}]\!], \rho\rangle\langle aft[\![\text{break;}]\!], \rho\rangle$$

که نشانه‌ی توقف است.

$\blacktriangleright\blacktriangleright$ S = $\{\mathsf{Sl''}\}$:

در این صورت، توقف پذیری $\mathsf{Sl''}$ را از فرض استقرای، استقرای قبلی(روی لیست عبارت‌های دستوری) داریم، پس $\{\mathsf{Sl''}\}$ هم توقف پذیر است.

در اینجا، اثبات این طرف قضیه به پایان می‌رسد.

($\Leftarrow$) دوباره باید از استقرا روی ساختار برنامه‌ها استفاده کنیم و دوباره چون هر برنامه‌ای مساوی با یک لیست از عبارت‌های دستوری است، ابتدا از استقرا روی ساختار لیست عبارت‌های و در دل آن روی ساختار خود عبارت‌های دستوری استفاده کنیم.

در این اثبات به غیر از یک حالت ساختار یک عبارت دستوری، که عبارت دستوری حلقه است، هر آنچه در مورد اثبات طرف راست قضیه گفتیم، به ما حکم را بدون نیاز به فرض نشان می‌دهد. بنابراین، فقط در مورد اثبات این یک حالت بحث می‌کنیم:

$$\blacktriangleright\blacktriangleright\ \mathsf{S} = \ \mathsf{while}\ (\mathsf{B})\ \mathsf{S_t}:$$

اگر فرض کنیم این عبارت دستوری به ازای محیط ρ در حالت اول متوقف شده است، در واقع فرض کرده‌ایم که در معنای این عبارت دستوری رد پیشوندی $\langle at[\![\mathsf{S}]\!], \rho\rangle\pi\langle aft[\![\mathsf{S}]\!], \rho'\rangle\pi'$ وجود دارد. باید ثابت کنیم، به ازای π' دلخواه، اگر رد پیشوندی $\langle at[\![\mathsf{S}]\!], \rho\rangle\pi\langle aft[\![\mathsf{S}]\!], \rho'\rangle\pi'$ داخل $\mathcal{S}^*[\![\mathsf{S}]\!]$ وجود داشته باشد، آنگاه $\pi' = \epsilon$ برقرار است.

اگر برچسب $aft[\![\mathsf{S}]\!]$ در یک وضعیت از رد پیشوندی‌ای که گفتیم، حضور داشته باشد، یعنی در یک دور از اجرای حلقه، عبارت بولی لزوما معنای غلط داشته است. این باعث شده است که یک وضعیت شامل برچسب aft به یک رد پیشوندی چسبانده بشود و درنتیجه، این رد پیشوندی ساخته شده است. از طرفی دیگر هم می‌دانیم که وقتی معنای عبارت بولی حاضر در ساختار حلقه غلط شده باشد، دیگر به ردهای پیشوندی داخل معنای حلقه چیزی اضافه نمی‌شود. بنابراین حالت ممکنی جز $\pi' = \epsilon$ باقی نمی‌ماند. $\square$

پس با توجه به آنچه گفتیم، می‌توانیم با خیال راحت توقف پذیری یک برنامه با یک محیط اولیه را معادل متناهی بودن همه‌ی ردهای پیشوندی‌ای بدانیم که با محیط متناظر با آن محیط اولیه شروع شده‌اند. اگر در صورت ارائه شده از وارسی مدل عبارت منظم R را با عبارت منظم ε جایگزین کنیم داریم:

$$\mathsf{P}, \underline{\rho} \models \mathsf{R} \Leftrightarrow (\{\underline{\rho}\} \times \mathcal{S}^*[\![\mathsf{P}]\!]) \subseteq \mathsf{prefix}(\mathcal{S}^r[\![\varepsilon \bullet (?:T)^*]\!]) = \mathsf{prefix}(\mathcal{S}^r[\![(?:T)^*]\!])$$

طبق تعریف معنای عبارات منظم، هر رد پیشوندی متناهی‌ای داخل مجموعه‌ی سمت راستی رابطه‌ی زیرمجموعه بودن قرار می‌گیرد. این یعنی اگر الگوریتمی برای بررسی $\mathsf{P}, \underline{\rho} \models \mathsf{R}$ داشته باشیم، این الگوریتم می‌تواند تشخیص دهد که آیا برنامه‌ی P با محیط اولیه‌ی $\underline{\rho}$ متوقف می‌شود یا خیر. این الگوریتمی است برای مسئله‌ی توقف پذیری، مسئله‌ای که تصمیم ناپذیر است. بنابراین، چنین الگوریتمی نباید وجود داشته باشد که یعنی پیاده سازی‌ای برای روشی که در حال بیانش هستیم وجود ندارد. ادامه‌ی کار روی همین تعریف پیش می‌رود و دو صورت دیگر هم که قرار است ساختارمندتر باشند، در نهایت با این صورت معادل‌اند و در نتیجه تصمیم ناپذیرند.

فصل ۳

وارسی مدل منظم

در این فصل، به صورتی ساختارمندتر برای روش وارسی مدل می‌رسیم. ساختاری که در این فصل به صورت روش وارسی مدل اضافه می‌شود، ساختار عبارات منظم است و نام این صورت هم وارسی مدل منظم[1] است. از این رو، پیش از اینکه به بیان وارسی مدل منظم بپردازیم، نیاز داریم که ابتدا، به بررسی و تعریف برخی خواص عبارات منظم بپردازیم که در ادامه، برای صورت وارسی مدل منظم مورد نیاز هستند.

۱.۳ در مورد عبارات منظم

در این بخش، ابتدا مفهوم هم‌ارز بودن را برای عبارت‌های منظم تعریف می‌کنیم، سپس به سراغ تعریف دو تابع dnf و fstnxt می‌رویم.

۱.۱.۳ هم‌ارزی عبارت‌های منظم

هم‌ارزی[2] بین دو عبارت منظم را با برابر بودن معنای آن دو تعریف می‌کنیم.

تعریف ۱.۳. (هم‌ارزی عبارات منظم): دو عبارت منظم R_1 و R_2 را هم‌ارز می‌گوییم، اگر و تنها

[1] Regular Model Checking
[2] Equivalence

اگر شرط زیر برقرار باشد:

$$\mathcal{S}^r\llbracket \mathsf{R}_1 \rrbracket = \mathcal{S}^r\llbracket \mathsf{R}_2 \rrbracket$$

این هم‌ارزی را با $\mathsf{R}_1 \Bumpeq \mathsf{R}_2$ نمایش می‌دهیم.

قضیه ۲.۳. هم‌ارزی $\Bumpeq$ تعریف شده روی مجموعه‌ی عبارت‌های منظم یک رابطه‌ی هم‌ارزی است.

اثبات. برای هر عبارت منظم R داریم:

$$\mathcal{S}^r\llbracket \mathsf{R} \rrbracket = \mathcal{S}^r\llbracket \mathsf{R} \rrbracket \Rightarrow \mathsf{R} \Bumpeq \mathsf{R}$$

پس این رابطه بازتابی است.
اگر $\mathsf{R}_1, \mathsf{R}_2 \in \mathbb{R}$ آنگاه داریم:

$$\mathcal{S}^r\llbracket \mathsf{R}_1 \rrbracket = \mathcal{S}^r\llbracket \mathsf{R}_2 \rrbracket \to \mathcal{S}^r\llbracket \mathsf{R}_2 \rrbracket = \mathcal{S}^r\llbracket \mathsf{R}_1 \rrbracket \Rightarrow \mathsf{R}_1 \Bumpeq \mathsf{R}_2 \to \mathsf{R}_2 \Bumpeq \mathsf{R}_1$$

پس این رابطه تقارنی هم هست.
اگر $\mathsf{R}_1, \mathsf{R}_2, \mathsf{R}_3 \in \mathbb{R}$ داریم:

$$\mathcal{S}^r\llbracket \mathsf{R}_1 \rrbracket = \mathcal{S}^r\llbracket \mathsf{R}_2 \rrbracket \wedge \mathcal{S}^r\llbracket \mathsf{R}_2 \rrbracket = \mathcal{S}^r\llbracket \mathsf{R}_3 \rrbracket \to \mathcal{S}^r\llbracket \mathsf{R}_1 \rrbracket = \mathcal{S}^r\llbracket \mathsf{R}_3 \rrbracket$$

$$\Rightarrow \mathsf{R}_1 \Bumpeq \mathsf{R}_2 \wedge \mathsf{R}_2 \Bumpeq \mathsf{R}_3 \to \mathsf{R}_1 \Bumpeq \mathsf{R}_3$$

پس این رابطه متعدی نیز هست.
پس این یک رابطه‌ی هم‌ارزی است. □

۳.۱.۲ فرم نرمال فصلی

یک دسته از عبارت‌های منظم هستند که به آن‌ها فرم نرمال فصلی[۳] می‌گوییم. در صورتی از وارسی مدل که در این فصل ارائه شده است، مفهوم فرم نرمال فصلی حضور دارد، بنابراین باید به بحث در مورد آن، پیش از رسیدن به صورت جدید، بپردازیم.

تعریف ۳.۳. (فرم نرمال فصلی): عبارت منظم $\mathsf{R} \in \mathbb{R}$ را یک فرم نرمال فصلی می‌گوییم، اگر و تنها اگر با فرض اینکه عبارت‌های منظم بدون انتخاب $\mathsf{R}_1, \mathsf{R}_2, ..., \mathsf{R}_n \in \mathbb{R}^{\nmid}$ وجود داشته باشند که $\mathsf{R} = \mathsf{R}_1 + \mathsf{R}_2 + ...\mathsf{R}_n$.

[3] Disjunctive Normal Form

در تعریف بالا، به = دقت شود که با $\Leftrightarrow$ که در ادامه مورد بحث ماست فرق می‌کند. منظور از = همان تساوی نحوی است.

در ادامه می‌خواهیم، یک تابع به اسم dnf تعریف کنیم که یک عبارت منظم R را می‌گیرد و عبارت منظم R' را تحویل می‌دهد. R' یک فرم نرمال فصلی است و $R \Leftrightarrow R'$ برقرار است. ابتدا، این تابع را به صورت استقرایی روی ساختار عبارات منظم تعریف می‌کنیم، سپس خاصیتی که گفتیم را در مورد آن ثابت می‌کنیم. این اثباتی بر این گزاره خواهد بود که هر عبارت منظم با یک فرم نرمال فصلی هم ارز است.

تعریف ۴.۳. (تابع dnf): تابع dnf روی عبارات منظم به شکل زیر تعریف می‌شود:

$$\blacktriangleleft\ \mathsf{dnf}(\varepsilon) = \varepsilon$$

$$\blacktriangleleft\ \mathsf{dnf}(L : B) = L : B$$

$$\blacktriangleleft\ \mathsf{dnf}(R_1R_2) = \Sigma_{i=1}^{n_1}\Sigma_{j=1}^{n_2}R_1^iR_2^j$$

$$\text{where } R_1^1 + R_1^2 + ... + R_1^{n_1} = \mathsf{dnf}(R_1) \text{ and } R_2^1 + R_2^2 + ... + R_2^{n_2} = \mathsf{dnf}(R_2)$$

$$\blacktriangleleft\ \mathsf{dnf}(R_1 + R_2) = \mathsf{dnf}(R_1) + \mathsf{dnf}(R_2)$$

$$\blacktriangleleft\ \mathsf{dnf}(R^*) = ((R_1)^*(R_2)^*...(R_n)^*)^*$$

$$\text{where } \mathsf{dnf}(R) = R^1 + R^2 + ... + R^n$$

$$\blacktriangleleft\ \mathsf{dnf}(R^+) = \mathsf{dnf}(RR^*)$$

$$\blacktriangleleft\ \mathsf{dnf}((R)) = (\mathsf{dnf}(R))$$

قضیه ۵.۳. اگر $R \in \mathbb{R}$ آنگاه $\mathsf{dnf}(R)$ یک ترکیب نرمال فصلی است.

اثبات. از استقرا روی ساختار R استفاده می‌کنیم.

$$\blacktriangleright\ R = \varepsilon :$$

$$\mathsf{dnf}(\varepsilon) = \varepsilon$$

که ε یک فرم نرمال فصلی است.

$$\blacktriangleright\ R = L : B :$$

$$dnf(L : B) = L : B$$

که $L : B$ هم یک فرم نرمال فصلی است.

$$\blacktriangleright\ R = R_1R_2 :$$

فرض استقرا این است که $dnf(R_1) = R_1^1 + R_1^2 + ... + R_1^n$ و $dnf(R_2) = R_2^1 + R_2^2 + ... + R_2^n$.
درحالیکه، $dnf(R_1)$ و $dnf(R_2)$ ترکیب نرمال فصلی هستند، یعنی هر R_1^i و هر R_2^j عضو $\mathbb{R}^\dagger$ است.
طبق تعریف، خواهیم داشت:

$$dnf(R_1R_2) = \Sigma_{i=1}^{n_1}\Sigma_{j=1}^{n_2}R_1^iR_2^j$$

که طرف راست گزاره‌ی بالا یک ترکیب نرمال فصلی است، چون هر $R_1^iR_2^j$ یک عضو از $\mathbb{R}^\dagger$ است.

$$\blacktriangleright\ R = R_1 + R_2 :$$

فرض استقرا این است که $dnf(R_1)$ و $dnf(R_2)$ ترکیب نرمال فصلی هستند. طبق تعریف، $dnf(R_1 + R_2)$ برابر با $dnf(R_1) + dnf(R_2)$ است. بنابراین، این عبارت منظم یک ترکیب نرمال فصلی است.

$$\blacktriangleright\ R = R_1^* :$$

طبق فرض استقرا، داریم که $dnf(R_1)$ یک ترکیب نرمال فصلی است. همین طور طبق تعریف dnf داریم:

$$dnf(R_1^*) = ((R_1^1)^*(R_1^2)^*...(R_1^n)^*)$$

که:

$$dnf(R_1) = R_1^1 + R_1^2 + ... + R_1^n$$

که اینکه $((R_1^1)^*(R_1^2)^*...(R_1^n)^*)$ یک فرم نرمال فصلی است، مشخص است، چون می‌دانیم که در هیچ کدام از این R_1^i ها عملگر $+$ وجود ندارد و عملگر $*$ و عملگر چسباندن هم تغییری در این وضع ایجاد نمی‌کنند.

$$\blacktriangleright\ R = R_1^+ :$$

طبق تعریف، داریم:

$$dnf(R_1^+) = dnf(R_1 R_1^*)$$

$$dnf(R_1^*) = ((R_1^1)^*(R_1^2)^* ... (R_1^n)^*)$$

که گیریم $R' = dnf(R_1^*)$ که عضو $\mathbb{R}^{\dagger}$ است. علاوه بر این، از فرض استقرا داریم:

$$R_1 = R_1^1 + ... + R_1^n$$

پس با توجه به تعریف dnf برای عملگر چسباندن و پخش پذیری چسباندن نسبت به انتخاب، خواهیم داشت:

$$dnf(R_1^+) = \Sigma_{i=1}^n R_1^i R'$$

$$\blacktriangleright\ R = (R_1) :$$

طبق تعریف، داریم:

$$dnf((R_1)) = (dnf(R_1))$$

طبق فرض استقرا $dnf(R_1)$ یک ترکیب نرمال فصلی است، بنابراین $(dnf(R_1)) = R' \in \mathbb{R}^{\dagger}$ هم یک ترکیب نرمال فصلی خواهد بود.

$\square$

گزاره‌ی دیگری که برای اثبات مانده است، برقرار بودن $dnf(R) \Bumpeq R$ است. برای اثبات آن باید ابتدا قضیه‌ی زیر را اثبات کنیم که اثبات آن را ارجاع می‌دهیم به [۱۳].

قضیه ۶.۳. برای هر دو عبارت منظم $R_1, R_2 \in \mathbb{R}$ داریم:

$$(R_1 + R_2)^* \Bumpeq (R_1^* R_2^*)^*$$

به عنوان نتیجه از قضیه‌ی بالا، می‌توانیم با استفاده از یک برهان ساده به کمک استقرا روی اعداد طبیعی، حکم بالا را به جای ۲ برای تعداد دلخواه متناهی‌ای از عبارات منظم اثبات کنیم. در ادامه در واقع از این حکم در اثبات استفاده شده است.

قضیه ۷.۳. برای هر $\mathsf{R} \in \mathbb{R}$ داریم:

$$\mathsf{dnf(R)} \doteqdot \mathsf{R}$$

اثبات. این اثبات با استقرا روی ساختار R انجام می‌شود. توجه شود که در هر حالت از استقرا اگر عبارات منظم $\mathsf{R_1, R_2}$ در ساختار R حضور داشته باشند، در مورد آن‌ها فرض گرفته‌ایم که $\mathsf{dnf(R_1) = R_1^1 + R_1^2 + ... + R_1^n}$ و $\mathsf{dnf(R_2) = R_1^2 + R_2^2 + ... + R_2^m}$.

$$\blacktriangleright\ \mathsf{R} = \varepsilon :$$

$$\mathsf{dnf}(\varepsilon) = \varepsilon \Rightarrow \mathcal{S}^r[\![\mathsf{dnf}(\varepsilon)]\!] = \mathcal{S}^r[\![\varepsilon]\!]$$

$$\blacktriangleright\ \mathsf{R = L : B} \ :$$

$$\mathsf{dnf(L : B) = L : B} \Rightarrow \mathcal{S}^r[\![\mathsf{dnf(L : B)}]\!] = \mathcal{S}^r[\![\mathsf{L : B}]\!]$$

$$\blacktriangleright\ \mathsf{R = R_1R_2} \ :$$

برای اثبات این حالت، باید دو گزاره‌ی زیر را ثابت کنیم:

$$\mathcal{S}^r[\![\mathsf{R_1R_2}]\!] \subseteq \mathcal{S}^r[\![\mathsf{dnf(R_1R_2)}]\!]$$

$$\mathcal{S}^r[\![\mathsf{R_1R_2}]\!] \supseteq \mathcal{S}^r[\![\mathsf{dnf(R_1R_2)}]\!]$$

$(\supseteq)$:

فرض می‌کنیم که $\langle \underline{\rho}, \pi \rangle$ یک عضو دلخواه از $\mathcal{S}^r[\![\mathsf{dnf(R_1R_2)}]\!]$ باشد. چون $\mathsf{dnf(R_1R_2)} = \Sigma_{\mathsf{i=1}}^{\mathsf{n_1}} \Sigma_{\mathsf{j=1}}^{\mathsf{n_2}} \mathsf{R_1^i R_2^j}$، پس داریم:

$$\exists k_1, k_2 : \pi \in \mathcal{S}^r[\![\mathsf{R_1^{k_1} R_2^{k_2}}]\!]$$

$$\Rightarrow \exists \pi_1, \pi_2 \ s.t. \ \pi = \pi_1\pi_2, \langle \underline{\rho}, \pi_1 \rangle \in \mathcal{S}^r[\![\mathsf{R_1^{k_1}}]\!], \langle \underline{\rho}, \pi_2 \rangle \in \mathcal{S}^r[\![\mathsf{R_2^{k_2}}]\!]$$

در این صورت، داریم:

$$\mathcal{S}^r[\![\mathsf{R_1^{k_1}}]\!] \subseteq \mathcal{S}^r[\![\mathsf{R_1}]\!], \mathcal{S}^r[\![\mathsf{R_2^{k_2}}]\!] \subseteq \mathcal{S}^r[\![\mathsf{R_2}]\!]$$

$$\Rightarrow \langle \underline{\rho}, \pi_1\pi_2 \rangle = \langle \underline{\rho}, \pi \rangle \in \mathcal{S}^r[\![\mathsf{R}_1\mathsf{R}_2]\!]$$

$(\subseteq)$:

$$\langle \underline{\rho}, \pi \rangle \in \mathcal{S}^r[\![\mathsf{R}_1\mathsf{R}_2]\!]$$

$$\Rightarrow \exists \pi_1, \pi_2 : \pi = \pi_1\pi_2 \ s.t. \ \langle \underline{\rho}, \pi_1 \rangle \in \mathcal{S}^r[\![\mathsf{R}_1]\!], \langle \underline{\rho}, \pi_2 \rangle \in \mathcal{S}^r[\![\mathsf{R}_2]\!]$$

طبق فرض استقرا، داریم:

$$\mathcal{S}^r[\![\mathsf{R}_1]\!] = \mathcal{S}^r[\![\mathsf{dnf}(\mathsf{R}_1)]\!], \mathcal{S}^r[\![\mathsf{R}_2]\!] = \mathcal{S}^r[\![\mathsf{dnf}(\mathsf{R}_2)]\!],$$

$$\Rightarrow \exists k_1, k_2 : \langle \underline{\rho}, \pi_1 \rangle \in \mathcal{S}^r[\![\mathsf{R}_1^{\mathsf{k}_1}]\!], \langle \underline{\rho}, \pi_2 \rangle \in \mathcal{S}^r[\![\mathsf{R}_2^{\mathsf{k}_2}]\!]$$

$$\Rightarrow \langle \underline{\rho}, \pi \rangle \in \mathcal{S}^r[\![\mathsf{R}_1^{\mathsf{k}_1}\mathsf{R}_2^{\mathsf{k}_2}]\!] \subseteq \mathcal{S}^r[\![\mathsf{dnf}(\mathsf{R}_1\mathsf{R}_2)]\!]$$

$$\blacktriangleright \ \mathsf{R} = \mathsf{R}_1 + \mathsf{R}_2 :$$

$$\mathcal{S}^r[\![\mathsf{dnf}(\mathsf{R}_1 + \mathsf{R}_2)]\!] =$$

$$\mathcal{S}^r[\![\mathsf{dnf}(\mathsf{R}_1) + \mathsf{dnf}(\mathsf{R}_2)]\!] =$$

$$\mathcal{S}^r[\![\mathsf{dnf}(\mathsf{R}_1)]\!] \cup \mathcal{S}^r[\![\mathsf{dnf}(\mathsf{R}_2)]\!] =$$

(به کمک فرض استقرا)

$$\mathcal{S}^r[\![\mathsf{R}_1]\!] \cup \mathcal{S}^r[\![\mathsf{R}_2]\!] =$$

$$\mathcal{S}^r[\![\mathsf{R}_1 + \mathsf{R}_2]\!]$$

$$\blacktriangleright \ \mathsf{R} = \mathsf{R}_1^* :$$

$$\mathcal{S}^r[\![\mathsf{dfn}(\mathsf{R}_1^*)]\!] =$$

$$\mathcal{S}^r[\![((\mathsf{R}_1^1)^*(\mathsf{R}_1^2)^*...(\mathsf{R}_1^{\mathsf{n}}))^*]\!] =$$

(طبق نتیجه‌ای که از قضیه‌ی قبل گرفتیم)

$$\mathcal{S}^r[\![(\mathsf{R}_1^1 + \mathsf{R}_1^2 + ... + \mathsf{R}_1^{\mathsf{n}})^*]\!] =$$

$$\mathcal{S}^r[\![(\mathsf{R}_1)^*]\!] =$$

$$\mathcal{S}^r[\![\mathsf{R}_1^*]\!]$$

$$\blacktriangleright\ \mathsf{R} = \mathsf{R}_1^+ :$$

$$\mathcal{S}^r[\![\mathsf{dfn}(\mathsf{R}_1^+)]\!] =$$

$$\mathcal{S}^r[\![\mathsf{dfn}(\mathsf{R}_1\mathsf{R}_1^*)]\!]$$

در اینجا، عملگر چسباندن را داریم. در حالت‌های قبلی، این را نشان دادیم که چه طور در حضور عملگر چسباندن، حکم برقرار می‌شود. همان اثبات را در مورد همین گزاره هم می‌توانیم، بگوییم و پس از آن، به این شکل ادامه دهیم:

$$\mathcal{S}^r[\![\mathsf{dfn}(\mathsf{R}_1\mathsf{R}_1^*)]\!] = \mathcal{S}^r[\![\mathsf{R}_1\mathsf{R}_1^*]\!] = \mathcal{S}^r[\![\mathsf{R}_1^+]\!]$$

$$\blacktriangleright\ \mathsf{R} = (\mathsf{R}_1) :$$

$$\mathcal{S}^r[\![\mathsf{dnf}((\mathsf{R}_1))]\!] =$$

$$\mathcal{S}^r[\![\mathsf{dnf}(\mathsf{R}_1)]\!] =$$

(طبق فرض استقرا:)

$$\mathcal{S}^r[\![\mathsf{R}_1]\!] =$$

$$\mathcal{S}^r[\![(\mathsf{R}_1)]\!]$$

$\square$

۳.۱.۳ سر و دم عبارات منظم

در این بخش، تابعی را روی عبارت‌های منظم تعریف می‌کنیم که یک عبارت منظم را می‌گیرد و یک زوج مرتب از عبارت‌های منظم را تحویل می‌دهد، سپس به بیان یک قضیه در مورد این تابع می‌پردازیم. این تابع را با fstnxt نشان می‌دهیم و نام آن سر و دم[۴] است. قرار است این تابع یک

[4] First Next

عبارت منظم را بگیرد و آن را به این شکل تجزیه کند که اولین اتم موجود در عبارت منظم که انگار سر عبارت منظم است، از باقی آن که دم آن عبارت منظم می‌شود، جدا شود. تابع روی عبارات منظم تهی و عبارات منظمی که عملگر $+$ را دارند تعریف نمی‌شود.

تعریف ۸.۳. (تابع سر و دم): تابع سر و دم را از نوع $\mathsf{fstnxt} : (\mathbb{R}^+ \cap \mathbb{R}^\dagger) \to \mathbb{R} \times \mathbb{R}$ به شکل زیر تعریف می‌کنیم:

$$\blacktriangleleft\ \mathsf{fstnxt}(\mathsf{L} : \mathsf{B}) = \langle \mathsf{L} : \mathsf{B}, \varepsilon \rangle$$

$$\blacktriangleleft\ \mathsf{fstnxt}(\mathsf{R}_1\mathsf{R}_2) = \mathsf{fstnxt}(\mathsf{R}_2)$$

$$\text{where } \mathsf{R}_1 \in \mathbb{R}_\varepsilon$$

$$\blacktriangleleft\ \mathsf{fstnxt}(\mathsf{R}_1\mathsf{R}_2) = (\!|\mathsf{R}_1^{\mathsf{n}} \in \mathbb{R}_\varepsilon\ ?\ \langle \mathsf{R}_1^{\mathsf{f}}, \mathsf{R}_2 \rangle : \langle \mathsf{R}_1^{\mathsf{f}}, \mathsf{R}_1^{\mathsf{n}} \bullet \mathsf{R}_2 \rangle |\!)$$

$$\text{where } \mathsf{R}_1 \notin \mathbb{R}_\varepsilon \text{and } \mathsf{fstnxt}(\mathsf{R}_1) = \langle \mathsf{R}_1^{\mathsf{f}}, \mathsf{R}_1^{\mathsf{n}} \rangle$$

$$\blacktriangleleft\ \mathsf{fstnxt}(\mathsf{R}^+) = (\!|\mathsf{R}^{\mathsf{n}} \in \mathbb{R}_\varepsilon\ ?\ \langle \mathsf{R}^{\mathsf{f}}, \mathsf{R}^* \rangle : \langle \mathsf{R}^{\mathsf{f}}, \mathsf{R}^{\mathsf{n}} \bullet \mathsf{R}^* \rangle |\!)$$

$$\text{where } \mathsf{fstnxt}(\mathsf{R}) = \langle \mathsf{R}^{\mathsf{f}}, \mathsf{R}^{\mathsf{n}} \rangle$$

$$\blacktriangleleft\ \mathsf{fstnxt}((\mathsf{R})) = \mathsf{fstnxt}(\mathsf{R})$$

از این تعریف در صورتی از وارسی مدل که در این فصل ارائه می‌شود، استفاده می‌شود. یک قضیه در آخر این بخش آمده است که مهم‌ترین نتیجه در مورد تابع سر و دم است. برای اثبات آن قضیه ابتدا یک نحو برای $\mathbb{R}^+ \cap \mathbb{R}^\dagger$ می‌آوریم.

قضیه ۹.۳. ساختار زیر نحو اعضای $\mathbb{R}^+ \cap \mathbb{R}^\dagger$ را توصیف می‌کند.

$$\mathsf{R} \quad \in \quad \mathbb{R}^+ \cap \mathbb{R}^\dagger$$

$$\mathsf{R} \quad ::= \quad \mathsf{L} : \mathsf{B} \mid \varepsilon\mathsf{R}_2 \mid \mathsf{R}_1\varepsilon \mid \mathsf{R}_1\mathsf{R}_2 \mid \mathsf{R}_1^+ \mid (\mathsf{R}_1)$$

اثبات. نام مجموعه‌ی عبارت‌های منظم تولید شده با ساختار بالا را $\mathbb{R}'$ می‌گذاریم. باید ثابت کنیم $\mathbb{R}' = \mathbb{R}^+ \cap \mathbb{R}^\dagger$ که در این راه مجبور هستیم که ثابت کنیم، این دو مجموعه زیر مجموعه‌ی یکدیگر هستند. برای اثبات $\mathbb{R}' \subseteq \mathbb{R}^+ \cap \mathbb{R}^\dagger$ می‌توانیم از استقرا روی ساختار بالا استفاده کنیم:

$$\blacktriangleright\ \mathsf{R} = \mathsf{L} : \mathsf{B}\ :$$

به وضوح $\mathbb{R}^\dagger$ $\mathsf{L} : \mathsf{B} \in \mathbb{R}^+ \cap \mathbb{R}^\dagger$ است و این را از ساختار $\mathbb{R}^+$ و $\mathbb{R}^\dagger$ می‌توانیم ببینیم.

$$\blacktriangleright\ \mathsf{R} = \varepsilon \mathsf{R}_2\ :$$

با فرض اینکه $\mathsf{R}_2 \in \mathbb{R}^+ \cap \mathbb{R}^\dagger$ که فرض استقراست، طبق ساختار $\mathbb{R}^+$ داریم $\varepsilon \mathsf{R}_2 \in \mathbb{R}^+$ و طبق ساختار $\mathbb{R}^\dagger$ داریم چون $\mathsf{R}_2 \in \mathbb{R}^\dagger$ و $\varepsilon \in \mathbb{R}^\dagger$، پس $\varepsilon \mathsf{R}_2 \in \mathbb{R}^\dagger$. پس داریم $\varepsilon \mathsf{R}_2 \in \mathbb{R}^+ \cap \mathbb{R}^\dagger$.

$$\blacktriangleright\ \mathsf{R} = \mathsf{R}_1 \varepsilon\ :$$

مشابه حالت قبل ثابت می‌شود.

$$\blacktriangleright\ \mathsf{R} = \mathsf{R}_1 \mathsf{R}_2\ :$$

طبق فرض استقرا، داریم $\mathsf{R}_1, \mathsf{R}_2 \in \mathbb{R}^\dagger \cap \mathbb{R}^+$ و در هر دو ساختار عملگر چسباندن را داریم. پس این حالت هم اثبات می‌شود.

$$\blacktriangleright\ \mathsf{R} = \mathsf{R}_1^+\ :$$

مثل حالت قبل، چون عملگر $^+$ در هر دو ساختار موجود است، به کمک فرض استقرا اثبات می‌شود.

$$\blacktriangleright\ \mathsf{R} = (\mathsf{R}_1)\ :$$

مثل حالت قبل اثبات می‌شود.

در اینجا اثبات $\mathbb{R}' \subseteq \mathbb{R}^+ \cap \mathbb{R}^\dagger$ کامل می‌شود. حال به سراغ اثبات $\mathbb{R}^+ \cap \mathbb{R}^\dagger \subseteq \mathbb{R}'$ می‌رویم.

برای اثبات این بخش، با فرض اینکه $\mathsf{R} \in \mathbb{R}^+ \cap \mathbb{R}^\dagger$، از استقرا روی ساختار اعضای $\mathbb{R}^\dagger$ استفاده می‌کنیم(این اثبات می‌توانست با استقرا روی ساختار اعضای $\mathbb{R}^+$ هم انجام شود).

$$\blacktriangleright\ \mathsf{R} = \varepsilon\ :$$

چون $\varepsilon \notin \mathbb{R}^+ \cap \mathbb{R}^\dagger$، پس این حالت باطل است و در مورد آن نیازی به ارائه‌ی اثبات نیست.

$$\blacktriangleright\ \mathsf{R} = \mathsf{L} : \mathsf{B}\ :$$

در این صورت، طبق ساختار $\mathbb{R}'$ داریم $\mathsf{R} \in \mathbb{R}'$.

$$\blacktriangleright\ \mathsf{R} = \mathsf{R}_1 \mathsf{R}_2\ :$$

اینجا هم با توجه به اینکه طبق فرض استقرا $R_1, R_2 \in \mathbb{R}'$، مثل حالت قبل چون $+$ در ساختار $\mathbb{R}'$ حضور دارد، حکم ثابت می‌شود.

$$\blacktriangleright\ R = R_1^* \ :$$

چون به ازای هیچ عبارت منظم R_1ای ${R_1}^*$ داخل $\mathbb{R}^+$ نمی‌افتد، پس بررسی این حالت هم مورد نیاز نیست.

$$\blacktriangleright\ R = R_1^+ \ :$$

مثل عملگر $+$، با توجه به فرض استقرا و اینکه $^+$ در ساختار $\mathbb{R}'$ حضور دارد، این حالت هم اثبات می‌شود.

$$\blacktriangleright\ R = (R_1) \ :$$

شبیه به حالت قبلی است.

□

ساختاری که تابع سر و دم روی آن تعریف شده است، با این ساختار ریخت متفاوتی دارد و البته لزومی هم ندارد که یکی باشند. ساختاری که در قضیه‌ی قبل ارائه کرده‌ایم، در [۶] نیامده است و خودمان با هدف اثبات قضیه‌ی بعدی، این ساختار را در اینجا ارائه کرده‌ایم.

قضیه ۳.۱۰. برای هر عبارت منظم $R \in \mathbb{R}^+ \cap \mathbb{R}^\dagger$ اگر $\mathrm{fstnxt}(R) = \langle L : B, R' \rangle$ آنگاه $R' \in \mathbb{R}^\dagger$ و $R \doteqdot L : B \bullet R'$.

اثبات. اثبات را باید با استفاده از استقرا روی ساختار عبارت‌های منظم عضو $\mathbb{R}^+ \cap \mathbb{R}^\dagger$ انجام داد.

$$\blacktriangleright\ R = L : B;$$

در این حالت، طبق تعریف تابع سر و دم، داریم $\mathrm{fstnxt}(R) = \langle L : B, \varepsilon \rangle$. از طرف دیگر $\mathcal{S}^r[\![L : B \bullet \varepsilon]\!] = \mathcal{S}^r[\![L : B]\!]$ را داریم و ε عضو $\mathbb{R}^\dagger$ است.

$$\blacktriangleright\ R = \varepsilon R_2;$$

طبق تعریف تابع سر و دم، داریم $\mathsf{fstnxt}(\varepsilon \mathsf{R}_2) = \mathsf{fstnxt}(\mathsf{R}_2)$. فرض استقرا این است که اگر $\mathsf{fstnxt}(\mathsf{R}_2) = \langle \mathsf{L} : \mathsf{B}, \mathsf{R}_2' \rangle$ آنگاه $\mathsf{R}_2' \in \mathbb{R}^{\dagger}$ و $\mathsf{R}_2 \doteqdot \mathsf{L} : \mathsf{B} \bullet \mathsf{R}_2'$. پس $\mathsf{fstnxt}(\mathsf{R}) = \langle \mathsf{L} : \mathsf{B}, \mathsf{R}_2' \rangle$ که همان طور که گفتیم طبق فرض استقرا $\mathsf{R}_2' \in \mathbb{R}^{\dagger}$ و از طرف دیگر:

$$\mathsf{R} = \varepsilon \mathsf{R}_2 \doteqdot \mathsf{R}_2 \doteqdot \mathsf{L} : \mathsf{B} \bullet \mathsf{R}_2'$$

$$\blacktriangleright\ \mathsf{R} = \mathsf{R}_1 \varepsilon;$$

در این حالت امکان ندارد $\mathsf{R}_1 = \varepsilon$ باشد، چون در آن صورت خواهیم داشت، $\mathsf{R} = \varepsilon\varepsilon \in \mathbb{R}_{\varepsilon}$ که تناقض است، چون $\varepsilon\varepsilon$ در دامنه‌ی تابع سر و دم نیست. طبق تعریف سر و دم اگر داشته باشیم $\mathsf{fstnxt}(\mathsf{R}_1) = \langle \mathsf{L} : \mathsf{B}, \mathsf{R}_1' \rangle$ در این صورت، بنا به اینکه $\mathsf{R}_1' \in \mathbb{R}_{\varepsilon}$ برقرار باشد یا نباشد، دو حالت را داریم:

$$\blacktriangleright\blacktriangleright\ \mathsf{R}_1' \in \mathbb{R}_{\varepsilon} :$$

در این صورت، $\mathsf{fstnxt}(\mathsf{R}) = \langle \mathsf{L} : \mathsf{B}, \varepsilon \rangle$ برقرار است. داریم $\mathsf{R} \in (\mathbb{R}^{+} \cap \mathbb{R}^{\dagger})$ پس چون R_2 زیر رشته‌ی R است، پس $\mathsf{R}_2 \in \mathbb{R}^{\dagger}$ برقرار است. در این صورت، $\mathsf{R} = \mathsf{L} : \mathsf{B}\ \varepsilon$ نیز برقرار خواهد بود. بنابراین، $\mathsf{L} : \mathsf{B}\ \varepsilon \doteqdot \mathsf{L} : \mathsf{B} \bullet \varepsilon$ هم بدیهی خواهد بود.

$$\blacktriangleright\blacktriangleright\ \mathsf{R}_1' \notin \mathbb{R}_{\varepsilon} :$$

در این صورت، گزاره‌ی $\mathsf{fstnxt}(\mathsf{R}) = \langle \mathsf{L} : \mathsf{B}, \mathsf{R}_1' \bullet \varepsilon \rangle$ طبق تعریف سر و دم برقرار است. چون $\mathsf{R} \in \mathbb{R}^{+} \cap \mathbb{R}^{\dagger}$، پس زیر رشته‌های آن نیز عملگر $+$ را نخواهند داشت، پس $\mathsf{R} \in \mathbb{R}^{\dagger}$. در این صورت نیز، واضح است که:

$$\mathsf{L} : \mathsf{B} \bullet \mathsf{R}_1' \bullet \varepsilon \doteqdot \mathsf{R}_1 \varepsilon = \mathsf{R}$$

$$\blacktriangleright\ \mathsf{R} = \mathsf{R}_1 \mathsf{R}_2;$$

اگر یکی از R_1 و R_2 مساوی ε باشند، حالاتی که بررسی کرده‌ایم اتفاق می‌افتند. اگر هر دو عبارت منظم برابر با ε باشند نیز به تناقض می‌خوریم، چون در این صورت دیگر در $\mathbb{R}^{+}$ این عبارت منظم را نداریم. پس تنها یک حالت می‌ماند و آن این است که هیچ یک از این دو عبارت منظم تهی

نباشند. در این صورت، اگر فرض کنیم $\mathsf{fstnxt}(\mathsf{R}_1) = \langle \mathsf{L} : \mathsf{B}, \mathsf{R}'_1 \rangle$، مسئله بنا به اینکه $\mathsf{R}'_1 \in \mathbb{R}_\varepsilon$ برقرار هست یا خیر، به دو حالت افراز می‌شود.

$$\blacktriangleright\blacktriangleright\ \mathsf{R}'_1 \notin \mathbb{R}_\varepsilon :$$

در این صورت $\mathsf{fstnxt}(\mathsf{R}) = \langle \mathsf{L} : \mathsf{B}, \mathsf{R}'_1 \bullet \mathsf{R}_2 \rangle$. مانند استدلالی که در حالت قبلی آوردیم، خواهیم داشت $\mathsf{R}'_1 \bullet \mathsf{R}_2 \in \mathbb{R}^\dagger$. علاوه‌بر این، طبق فرض استقرا داریم $\mathsf{L} : \mathsf{B} \bullet \mathsf{R}'_1 \doteqdot \mathsf{R}_1$، پس:

$$\mathsf{L} : \mathsf{B} \bullet \mathsf{R}'_1 \bullet \mathsf{R}_2 \doteqdot \mathsf{R}_1 \bullet \mathsf{R}_2 = \mathsf{R}$$

(عملگر چسباندن شرکت پذیر است). پس این حالت اثبات می‌شود.

$$\blacktriangleright\blacktriangleright\ \mathsf{R}'_1 \in \mathbb{R}_\varepsilon :$$

در این صورت، $\mathsf{fstnxt}(\mathsf{R}) = \langle \mathsf{L} : \mathsf{B}, \mathsf{R}_2 \rangle$. مثل حالت‌های قبل ثابت می‌شود که $\mathsf{R}_2 \in \mathbb{R}^\dagger$ و داریم:

$$\mathsf{R} = \mathsf{L} : \mathsf{B} \bullet \mathsf{R}_2 \Rightarrow \mathsf{L} : \mathsf{B} \bullet \mathsf{R}_2 \doteqdot \mathsf{R}$$

$$\blacktriangleright\ \mathsf{R} = \mathsf{R}_1^+;$$

با فرض اینکه $\mathsf{fstnxt}(\mathsf{R}_1) = \langle \mathsf{L} : \mathsf{B}, \mathsf{R}'_1 \rangle$، بنا به اینکه $\mathsf{R}'_1 \in \mathbb{R}_\varepsilon$ برقرار باشد یا نباشد، دو حالت خواهیم داشت:

$$\blacktriangleright\blacktriangleright\ \mathsf{R}'_1 \in \mathbb{R}_\varepsilon :$$

در این صورت طبق تعریف تابع سر و دم، $\mathsf{fstnxt}(\mathsf{R}) = \langle \mathsf{L} : \mathsf{B}, \mathsf{R}_1^* \rangle$ را داریم. چون داخل R'_1 و $\mathsf{L} : \mathsf{B}$ عملگر $+$ وجود ندارد R_1^* عضو $\mathbb{R}^\dagger$ خواهد بود. جای دیگری از این عبارت منظم وجود ندارد که در آن بتوان وجود این عملگر را متصور شد. همین طور، داریم:

$$\mathsf{fstnxt}(\mathsf{R}_1) = \langle \mathsf{L} : \mathsf{B}, \mathsf{R}'_1 \rangle \rightarrow \mathsf{R}'_1 \in \mathbb{R}^\dagger \ \wedge\ \mathsf{L} : \mathsf{B} \bullet \mathsf{R}'_1 \doteqdot \mathsf{R}_1$$

(گزاره‌ی بالا فرض استقراست.)

$$\mathsf{R}_1^* \doteqdot (\mathsf{L} : \mathsf{B} \bullet \mathsf{R}'_1)^* \doteqdot (\mathsf{L} : \mathsf{B} \bullet \varepsilon)^* \doteqdot (\mathsf{L} : \mathsf{B})^*$$

(هم‌ارزی وسطی به خاطر این است که R_1' عضو $\mathbb{R}^{\nmid}$ است. اگر یکی از دو هم‌ارزی دیگر هم برقرار نباشند کلا عملگر * خوش تعریف نخواهد بود، پس این دو هم‌ارزی باید برقرار باشند.)

$$\Rightarrow L : B \bullet R_1^* \simeq L : B \bullet (L : B)^* \simeq (L : B)^+$$

$$\blacktriangleright\blacktriangleright\ R_1' \notin \mathbb{R}_\varepsilon :$$

با توجه به تعریف تابع سر و دم و فرض استقرا که پیش‌تر بیان کرده‌ایم، در این حالت داریم $\mathrm{fstnxt}(R) = \langle L : B, R_1' \bullet R_1^* \rangle$. به همان دلیل حالت قبلی، می‌دانیم که R_1^* عضو $\mathbb{R}^{\nmid}$ است و طبق فرض استقرا داریم $R_1' \in \mathbb{R}^{\nmid}$. بنابراین داریم $R_1' \bullet R_1^* \in \mathbb{R}^{\nmid}$.

با استفاده از فرض استقرا داریم:

$$L : B \bullet R_1' \simeq R_1$$

$$\Rightarrow L : B \bullet R_1' \bullet R_1^* \simeq R_1 R_1^* \simeq R_1^+$$

$$\blacktriangleright\ R = (R_1)\ :$$

از فرض استقرا نتیجه می‌شود. $\square$

این بخش، در این قسمت به پایان می‌رسد. حال ابزارهای کافی برای بیان صورت وارسی مدل منظم را در اختیار داریم.

۲.۳ وارسی مدل منظم

همان‌طور که پیش‌تر گفتیم، می‌خواهیم در این فصل یک صورت معادل با صورتی که در فصل پیش برای روش وارسی مدل آورده شده بود، ارائه کنیم. تا اینجای این فصل، صرفا به معرفی چند مفهوم که برای بیان صورت جدید به آن‌ها احتیاج داریم، پرداخته‌ایم. در این یخش ابتدا این صورت جدید را بیان می‌کنیم و سپس اثبات می‌کنیم که صورت جدید با صورت قبلی معادل است. همان‌طور که پیش‌تر هم اشاره شد، تفاوت این دو صورت در این است که در این صورت ساختار عبارات منظم اثر دارد، در حالیکه، صورت قبلی ساختاری نداشت.

۱.۲.۳ صورت

در نهایت، برای تعریف صورت به یک تابع به نام $\mathcal{M}$ خواهیم رسید که در ورودی‌اش، یک زوج متشکل از یک محیط اولیه و یک عبارت منظم را در کنار یک برنامه می‌گیرد و در خروجی، همه‌ی ردهای پیشوندی موجود در معنای برنامه که عبارت منظم را ارضا می‌کنند، داخل یک مجموعه بر می‌گرداند. اما در این بین، مفهوم سازگاری یک رد پیشوندی با یک عبارت منظم چگونه مشخص می‌شود؟ این نکته‌ای است که تا به حال در مورد آن بحث نکرده‌ایم و حالا می‌خواهیم، تعریف تابع $\mathcal{M}^t$ را با این هدف به بحث وارد کنیم. البته، این تابع یک ویژگی بیشتر هم دارد. ویژگی دیگر است که اگر عبارت منظم با رد پیشوندی سازگار نباشد، تابع به ما می‌گوید که کجای عبارت منظم ناسازگاری وجود داشته است. همین طور اگر رد پیشوندی با عبارت منظم سازگار باشد، این تابع به ما نشان می‌دهد که عبارت منظم تا کجا بررسی شده است. فهمیدن این موضوع با نگاه به تعریف ساده‌تر است و البته، نباید فراموش کرد که سازگاری‌ای که داریم، هماهنگ با صورت قبلی است که در آن عملگر prefix و الحاق عبارت منظم $(? : T)^*$ را داشتیم.

تعریف ۱۱.۳. (وارسی‌گر رد پیشوندی): به تابع $\mathcal{M}^t$ از نوع $(\underline{\mathbb{EV}} \times \mathbb{R}^{\dagger}) \to \mathfrak{S}^{+\infty} \to (\mathbb{B} \times \mathbb{R}^{\dagger})$ وارسی‌گر رد پیشوندی می‌گوییم. این تابع ضابطه‌ی زیر را دارد:

$$\blacktriangleleft\ \mathcal{M}^t\langle \underline{\rho}, \varepsilon\rangle\pi = \langle T, \varepsilon\rangle$$

(برای هر عضو دیگر $\mathbb{R}_{\varepsilon}$ هم حالت بالایی برقرار است. دو حالت پایینی برای عبارت‌های منظم عضو $\mathbb{R}^{+} \cap \mathbb{R}^{\dagger}$ هستند.)

$$\blacktriangleleft\ \mathcal{M}^t\langle \underline{\rho}, \mathsf{R}\rangle\epsilon = \langle T, \mathsf{R}\rangle$$

$$\blacktriangleleft\ \mathcal{M}^t\langle \underline{\rho}, \mathsf{R}\rangle\pi = (\!|\langle \underline{\rho}, \langle l_1, \rho_1\rangle\rangle \in \mathcal{S}^r[\![\mathsf{L} : \mathsf{B}]\!]\ ?\ \mathcal{M}^t\langle \underline{\rho}, \mathsf{R}'\rangle\pi'\ :\ \langle F, \mathsf{R}\rangle|\!)$$

$$\text{where } \pi = \langle l_1, \rho_1\rangle\pi' \text{ and } \langle \mathsf{L} : \mathsf{B}, \mathsf{R}'\rangle = \mathsf{fstnxt}(\mathsf{R})$$

در مسیر رسیدن به تعریف $\mathcal{M}$، به معرفی یک تابع دیگر هم می‌پردازیم. این تابع را با $\mathcal{M}^{\dagger}$ نشان می‌دهیم. در واقع، همان کاری را که $\mathcal{M}$ قرار است به ازای همه‌ی عبارت‌های منظم انجام دهد، این تابع روی عبارت‌های منظمی که $+$ ندارند انجام می‌دهد.

تعریف ۱۲.۳. (وارسی مدل منظم محدود به $\mathbb{R}^{\dagger}$): به تابع $\mathcal{M}^{\dagger}$ از نوع $(\underline{\mathbb{EV}} \times \mathbb{R}^{\dagger}) \to P(\mathfrak{S}^{+\infty}) \to P(\mathfrak{S}^{+\infty} \times \mathbb{R}^{\dagger})$ می‌گوییم وارسی مدل منظم محدود به $\mathbb{R}^{\dagger}$. ضابطه‌ی این تابع به شکل زیر است:

$$\mathcal{M}^{\dagger}\langle \underline{\rho}, \mathsf{R}\rangle \Pi = \{\langle \pi, \mathsf{R}'\rangle | \pi \in \Pi \wedge \mathcal{M}^{t}\langle \underline{\rho}, \mathsf{R}\rangle \pi = \langle T, \mathsf{R}'\rangle\}$$

حالا خود $\mathcal{M}$ را تعریف می‌کنیم. تعریف این تابع چیزی نیست جز اجتماع گرفتن از خروجی تابع بالا، به ازای عبارات منظمی که در فرم نرمال فصلی عبارت منظم ورودی تابع حضور دارند. البته، بخشی از اطلاعاتمان از هر رد پیشوندی در هر زوجی که در خروجی $\mathcal{M}^{\dagger}$ وجود دارد، حذف می‌شود. به عبارت دیگر، صرفا ردهای پیشوندی را در مجموعه‌ای که خروجی $\mathcal{M}$ است، داریم. اطلاعات برای هر رد پیشوندی یک عبارت منظم است که بخشی از R است که سازگاری‌اش با رد پیشوندی بررسی نشده است. برای ردهای پیشوندی‌ای که در خروجی $\mathcal{M}$ حضور دارند و طولشان بیشتر یا مساوی عبارت منظم مورد بررسی است، این عبارت منظم برابر با تهی است.

تعریف ۱۳.۳. (وارسی مدل منظم):
تابع $\mathcal{M}$ را از نوع $(\underline{\mathbb{EV}} \times \mathbb{R}) \rightarrow P(\mathfrak{S}^{+\infty}) \rightarrow (\underline{\mathbb{EV}} \times P(\mathfrak{S}^{+\infty}))$ وارسی مدل منظم می‌گوییم که ضابطه‌ی زیر را دارد:

$$\mathcal{M}\langle \underline{\rho}, \mathsf{R}\rangle \Pi = \bigcup_{i=1}^{n} \{\langle \underline{\rho}, \pi\rangle | \exists \mathsf{R}' \in \mathbb{R} : \ \langle \pi, \mathsf{R}'\rangle \in \mathcal{M}^{\dagger}\langle \underline{\rho}, \mathsf{R_i}\rangle \Pi\}$$

$$\text{where } \mathsf{dnf}(\mathsf{R}) = \mathsf{R_1} + \mathsf{R_2} + \ldots + \mathsf{R_n}$$

در این صورت، اگر ویژگی $\mathsf{R} \in \mathbb{R}$ در محیط اولیه‌ی $\underline{\rho}$ برای برنامه‌ی P برقرار باشد، می نویسیم

$$\mathsf{P}, \underline{\rho} \models_r \mathsf{R}$$

و برقرار بودن این رابطه با شرط زیر تعریف می‌شود:

$$\mathsf{P}, \underline{\rho} \models_r \mathsf{R} \iff \{\underline{\rho}\} \times \mathcal{S}^{*}[\![\mathsf{P}]\!] \subseteq \mathcal{M}\langle \underline{\rho}, \mathsf{R}\rangle \mathcal{S}^{*}[\![\mathsf{P}]\!]$$

با این تعریف، در واقع زمانی می‌توانیم بگوییم، برنامه‌ی P ویژگی R دارد که $\mathcal{M}\langle \underline{\rho}, \mathsf{R}\rangle \mathcal{S}^{*}[\![\mathsf{P}]\!] = \{\underline{\rho}\} \times \mathcal{S}^{*}[\![\mathsf{P}]\!]$. در واقع، انگار تابع $\mathcal{M}\langle \underline{\rho}, \mathsf{R}\rangle$ مثل یک صافی روی مجموعه‌ی $\mathcal{S}^{*}[\![\mathsf{P}]\!]$ عمل می‌کند، پس خروجی این مجموعه زیر مجموعه‌ی $\{\underline{\rho}\} \times \mathcal{S}^{*}[\![\mathsf{P}]\!]$ است.

قضیه ۱۴.۳. برای هر برنامه‌ی P، محیط اولیه‌ی $\underline{\rho}$ و عبارت منظم R داریم:

$$\mathcal{M}\langle \underline{\rho}, \mathsf{R}\rangle \mathcal{S}^{*}[\![\mathsf{P}]\!] \subseteq \{\underline{\rho}\} \times \mathcal{S}^{*}[\![\mathsf{P}]\!]$$

اثبات. اگر زوج $\langle \underline{\rho}, \pi \rangle$ عضو $\mathcal{M}\langle \underline{\rho}, \mathsf{R} \rangle \mathcal{S}^*[\![\mathsf{P}]\!]$ باشد، با فرض $\mathsf{dnf}(\mathsf{R}) = \mathsf{R_1} + \mathsf{R_2} + \ldots + \mathsf{R_n}$ طبق تعریف $\mathcal{M}$، وجود دارند $\mathsf{R}' \in \mathbb{R}$ و عدد i بین ۱ و n که:

$$\langle \pi, \mathsf{R}' \rangle \in \mathcal{M}^{\dagger}\langle \underline{\rho}, \mathsf{R_i} \rangle \mathcal{S}^*[\![\mathsf{P}]\!]$$

که طبق تعریف $\mathcal{M}^{\dagger}$ یعنی:

$$\pi \in \mathcal{S}^*[\![\mathsf{P}]\!] \wedge \mathcal{M}^t\langle \underline{\rho}, \mathsf{R_i} \rangle \pi = \langle T, \mathsf{R}' \rangle$$

که از $\pi \in \mathcal{S}^*[\![\mathsf{P}]\!]$ می‌توان نتیجه گرفت $\langle \underline{\rho}, \pi \rangle \in \{\underline{\rho}\} \times \mathcal{S}^*[\![\mathsf{P}]\!]$.

□

مجموعه‌ی معنای یک برنامه را می‌توان به مجموعه‌ای از دسته‌ها افراز کرد که در هر یک از این دسته‌ها ردهای پیشوندی‌ای حضور دارند که وضعیت اول آن‌ها یکسان است. قاعدتا، در هر یک از این دسته‌ها باید وضعیت‌های بعدی هم، در صورت وجود، به طور موازی با یکدیگر یکسان باشند، یعنی مثلا در یک مجموعه از افراز توصیف شده، همه‌ی ردهای پیشوندی‌ای که عضو دوم دارند، عضو دومشان با هم برابر است. این گزاره در مورد عضو سوم و چهارم و غیره هم برقرار است. در هر دسته از این افراز، یک رد پیشوندی ماکسیمال[5] وجود خواهد داشت که توصیف تمام و کمال برنامه در اجرا با وضعیت اول مختص آن دسته است. اگر ردهای پیشوندی با محیط اولیه‌ی یکسان به اشکال مختلفی ادامه پیدا کنند، باید زبان برنامه نویسی مورد بررسی غیرقطعی باشد. در صورتیکه، در زبان و معناشناسی این زبانی که در فصل اول تعریف کردیم، مولفه‌ای از غیرقطعی بودن حضور ندارد.

حال برای هر برنامه‌ی P که ویژگی R در مورد آن در حال بررسی است، می‌توانیم همین افراز را روی مجموعه‌ی $\mathcal{M}\langle \underline{\rho}, \mathsf{R} \rangle \mathcal{S}^*[\![\mathsf{P}]\!]$ در نظر بگیریم. می‌توانیم هر دسته از این افراز را متناظر با دسته‌ای در افرازی که روی $\mathcal{S}^*[\![\mathsf{P}]\!]$ توصیف کردیم بدانیم، اگر و تنها اگر وضعیت اولیه در ردهای پیشوندی موجود در دو دسته یکسان باشند.

اگر در هر دسته از این افراز روی $\mathcal{S}^*[\![\mathsf{P}]\!]$، رد پیشوندی ماکسیمال این دسته در دسته‌ی متناظر در $\mathcal{M}\langle \underline{\rho}, \mathsf{R} \rangle \mathcal{S}^*[\![\mathsf{P}]\!]$ وجود داشته باشد، در این صورت پس حتما $\mathcal{M}\langle \underline{\rho}, \mathsf{R} \rangle \mathcal{S}^*[\![\mathsf{P}]\!] = \{\underline{\rho}\} \times \mathcal{S}^*[\![\mathsf{P}]\!]$. اگر دسته‌ای از افراز روی $\mathcal{M}\langle \underline{\rho}, \mathsf{R} \rangle \mathcal{S}^*[\![\mathsf{P}]\!]$ دارای عضو ماکسیمال متفاوتی نسبت به همان دسته روی $\mathcal{S}^*[\![\mathsf{P}]\!]$ باشد، قطعا این عضو ماکسیمال کوتاه‌تر از

[5] Maximal

عضو ماکسیمال همان دسته در $\mathcal{S}^*[\![\mathsf{P}]\!]$ است و این یعنی ناسازگاری‌ای با عبارت منظم مورد بررسی وجود داشته است. محل وقوع ناسازگاری را تابع $\mathcal{M}^\dagger$ می‌تواند به ما بگوید. محل ناسازگاری عبارت منظمی است که زوج عضو ماکسیمال دسته‌ی مورد نظر در خروجی یکی از اعمال های $\mathcal{M}^\dagger$ روی بخش‌های مختلف $\mathsf{dnf}(\mathsf{R})$ است.

۲.۲.۳ درستی و تمامیت

حال به اثبات معادل بودن صورت جدید با صورت قبلی می‌پردازیم. در [۶] این اثبات که یک قضیه‌ی دوطرفه است، تحت دو قضیه به نام‌های درستی[6] و تمامیت[7] آمده است. درستی به این معناست که اگر یک بررسی در صورت جدید انجام شود، نتیجه‌ای یکسان با انجام بررسی برای همان برنامه و همان عبارت منظم در صورت قبلی دارد. تمامیت نیز عکس درستی است، یعنی هر بررسی‌ای که با صورت قبلی انجام شده، نتیجه‌ی یکسانی با انجام همان بررسی در صورت جدید دارد.

نگارنده‌ی این پایان نامه، به درستی دو اثبات موجود در [۶] بسیار بد بین است! در اثبات تمامیت، برهان به شکل عجیبی بی‌ربط است و در اثبات قضیه درستی، ایرادات فنی ریزی در جزئیات وجود دارد که با تعاریف در تناقض است. از این رو برهان‌هایی که در اینجا آورده‌ایم، جدید هستند.

قضیه ۱۵.۳. (قضیه درستی): اگر P یک برنامه، R یک عبارت منظم و $\underline{\rho}$ یک محیط اولیه باشند، آنگاه داریم:

$$\mathsf{P}, \underline{\rho} \models_r \mathsf{R} \Rightarrow \mathsf{P}, \underline{\rho} \models \mathsf{R}$$

اثبات. طبق تعریف دو صورت، باید با فرض اینکه داریم:

$$\{\underline{\rho}\} \times \mathcal{S}^*[\![\mathsf{P}]\!] \subseteq \mathcal{M}\langle \underline{\rho}, \mathsf{R}\rangle \mathcal{S}^*[\![\mathsf{P}]\!]$$

ثابت کنیم:

$$\{\underline{\rho}\} \times \mathcal{S}^*[\![\mathsf{P}]\!] \subseteq \mathsf{prefix}(\mathcal{S}^{\mathsf{r}}[\![\mathsf{R} \bullet (? : T)^*]\!])$$

[6] Soundness

[7] Completeness

در این راستا، می‌توانیم گزاره‌ی زیر را ثابت کنیم که از گزاره‌ی قبلی قوی‌تر است و آن را نتیجه می‌دهد:

$$\mathcal{M}\langle \underline{\rho}, \mathsf{R}\rangle \mathcal{S}^*[\![\mathsf{P}]\!] \subseteq \mathsf{prefix}(\mathcal{S}^{\mathsf{r}}[\![\mathsf{R} \bullet (?:T)^*]\!])$$

در مورد عبارت منظم R، فرض می‌کنیم $\mathsf{dnf}(\mathsf{R}) = \mathsf{R}_1 + \mathsf{R}_2 + ... + \mathsf{R}_\mathsf{n}$، که هر R_i ذکر شده عضو $\mathbb{R}^\dagger$ است. فرض می‌کنیم $\langle \underline{\rho}, \pi\rangle \in \mathcal{M}\langle \underline{\rho}, \mathsf{R}\rangle \mathcal{S}^*[\![\mathsf{P}]\!]$، آنگاه وجود دارد k و R' که $\langle \pi, \mathsf{R}'\rangle \in \mathcal{M}^\dagger\langle \underline{\rho}, \mathsf{R}_k\rangle \mathcal{S}^*[\![\mathsf{P}]\!]$ که طبق تعریف $\mathcal{M}^\dagger$ خواهیم داشت:

$$\pi \in \mathcal{S}^*[\![\mathsf{P}]\!] \wedge \mathcal{M}^t\langle \underline{\rho}, \mathsf{R}_k\rangle \pi = \langle T, \mathsf{R}'\rangle$$

طرف چپ گزاره‌ی عطفی بالا در فرض بود. در ادامه‌ی کار با طرف راست این گزاره پیش می‌رویم:

$$\mathcal{M}^t\langle \underline{\rho}, \mathsf{R}_k\rangle \pi = \langle T, \mathsf{R}'\rangle$$

در مورد R_k دو حالت داریم، یا $\mathsf{R}_k \doteqdot \varepsilon$ برقرار است، یا اینگونه نیست($\mathsf{R}_k \not\doteqdot \varepsilon$).

$$\blacktriangleright\ \mathsf{R}_k \doteqdot \varepsilon :$$

در این صورت می‌توانیم، ثابت کنیم:

$$\mathsf{prefix}(\mathcal{S}^{\mathsf{r}}[\![\mathsf{R} \bullet (?:T)^*]\!]) = \{\underline{\rho}\} \times \mathfrak{S}^+$$

با توجه به پخش پذیری عملگر چسباندن روی عملگر انتخاب، که پیش‌تر ثابت کردیم، داریم:

$$\mathsf{R} \bullet (?:T)^* \doteqdot (\mathsf{R}_1 + \mathsf{R}_2 + ... + \mathsf{R}_\mathsf{m}) \bullet (?:T)^*$$

$$\doteqdot \mathsf{R}_1 \bullet (?:T)^* + \mathsf{R}_2 \bullet (?:T) + ... + \mathsf{R}_\mathsf{n} \bullet (?:T)^*$$

چون $\mathsf{R}_k \doteqdot \varepsilon$، داریم:

$$\mathsf{R}_1 \bullet (?:T)^* + \mathsf{R}_2 \bullet (?:T) + ... + \mathsf{R}_k \bullet (?:T)^* + ... + \mathsf{R}_\mathsf{n} \bullet (?:T)^*$$

$$\doteqdot \mathsf{R}_1 \bullet (?:T)^* + \mathsf{R}_2 \bullet (?:T) + ... + \varepsilon \bullet (?:T)^* + ... + \mathsf{R}_\mathsf{n} \bullet (?:T)^*$$

و از طرف دیگر داریم:

$$\varepsilon \bullet (?:T)^* \doteqdot (?:T)^* = (\{\underline{\rho}\} \times \mathfrak{S}^+)$$

پس $\mathsf{prefix}(\mathcal{S}^{\mathsf{r}}[\![\mathsf{R} \bullet (? : T)^*]\!])$ مجموعه‌ی $\{\underline{\rho}\} \times \mathfrak{S}^+$ را به عنوان زیرمجموعه در درون خود دارد و عضوی بیش از این هم طبق تعریفش نمی‌تواند داشته باشد، پس:

$$\mathsf{prefix}(\mathcal{S}^{\mathsf{r}}[\![\mathsf{R} \bullet (? : T)^*]\!]) = \{\underline{\rho}\} \times \mathfrak{S}^+$$

که این گزاره نتیجه می‌دهد:

$$\langle \underline{\rho}, \pi \rangle \in \mathsf{prefix}(\mathcal{S}^{\mathsf{r}}[\![\mathsf{R} \bullet (? : T)^*]\!])$$

► $\mathsf{R}_k \not\approx \varepsilon$:

همان طور که پیش‌تر اشاره کردیم، این فرض یعنی $\mathsf{R}_k \in \mathsf{R}^+ \cap \mathsf{R}^\dagger$. پس مجاز هستیم از تابع سر و دم استفاده کنیم. فرض می‌کنیم $\mathsf{fstnxt}(\mathsf{R}_k) = \langle \mathsf{L}_k^1 : \mathsf{B}_k^1, \mathsf{R}_k^1 \rangle$. همین طور فرض می‌کنیم:

$$\pi = \langle l_0, \rho_0 \rangle \langle l_1, \rho_1 \rangle \langle l_2, \rho_2 \rangle ... \langle l_l, \rho_l \rangle$$

و تعریف می‌کنیم:

$$\pi(i) = \langle l_i, \rho_i \rangle \langle l_{i+1}, \rho_{i+1} \rangle, ..., \langle l_l, \rho_l \rangle$$

داریم:

$$\mathcal{M}^t \langle \underline{\rho}, \mathsf{R}_k \rangle \pi = \langle T, \mathsf{R}' \rangle \Rightarrow \forall \mathsf{R}'' \in \mathbb{R} : \mathcal{M}^t \langle \underline{\rho}, \mathsf{R}_k \rangle \pi \neq \langle F, \mathsf{R}'' \rangle$$

پس لاجرم تساوی زیر برقرار است(با توجه به سر و دم R_k):

$$\mathcal{M}^t \langle \underline{\rho}, \mathsf{R}_k \rangle \pi = \mathcal{M}^t \langle \underline{\rho}, \mathsf{R}_k^1 \rangle \pi(1)$$

بدون کاستن از کلیت(چون ممکن است $\mathsf{R}_k^1 \approx \varepsilon$)، فرض می‌کنیم کاری که انجام دادیم را می‌توانیم روی دم خروجی عبارت منظم R_k(یعنی R_k^1) تکرار کنیم:

$$\mathcal{M}^t \langle \underline{\rho}, \mathsf{R}_k^1 \rangle \pi(1) = \mathcal{M}^t \langle \underline{\rho}, \mathsf{R}_k^2 \rangle \pi(2) \quad \text{where } \mathsf{fstnxt}(\mathsf{R}_k^1) = \langle \mathsf{L}_k^2 : \mathsf{B}_k^2, \mathsf{R}_k^2 \rangle$$

باز هم بدون کاستن از کلیت، می‌توانیم فرض کنیم که این رویه را به صورت یک سلسله می‌توان تا h مرحله ادامه داد، یعنی:

$$\mathcal{M}^t \langle \underline{\rho}, \mathsf{R}_k^{h-1} \rangle \pi(h-1) = \mathcal{M}^t \langle \underline{\rho}, \mathsf{R}_k^h \rangle \pi(h) \quad \text{where } \mathsf{fstnxt}(\mathsf{R}_k^{h-1}) = \langle \mathsf{L}_k^h : \mathsf{B}_k^h, \mathsf{R}_k^h \rangle$$

در حالیکه، $\mathsf{R}^\mathsf{h}_\mathsf{k} \doteqdot \varepsilon$. اگر گزاره‌ی $\mathsf{R}^\mathsf{h}_\mathsf{k} \doteqdot \varepsilon$ هیچ زمانی برقرار نشود، یعنی بتوانیم این سلسله را تا بی‌نهایت ادامه دهیم، مطمئن خواهیم بود که در معنای R_k حتما ردهای پیشوندی نامتناهی حضور دارند. چنین چیزی با تعریف معنای عبارات منظم در تناقض است، چون در معنای عبارات منظم رد پیشوندی نامتناهی حضور ندارد. تا اینجا، می‌توانیم بگوییم:

$$\mathsf{R}_\mathsf{k} \doteqdot \mathsf{L}^1_\mathsf{k} : \mathsf{B}^1_\mathsf{k} \bullet \mathsf{L}^2_\mathsf{k} : \mathsf{B}^2_\mathsf{k} \bullet \ldots \bullet \mathsf{L}^\mathsf{h}_\mathsf{k} : \mathsf{B}^\mathsf{h}_\mathsf{k}$$

حال بسته به اینکه $h < l$ برقرار باشد یا نباشد، می‌توانیم مسئله را به دو حالت افراز کنیم:

$$\blacktriangleright\blacktriangleright\ h < l :$$

در این صورت، داریم:

$$\mathcal{M}^t\langle \underline{\rho}, \mathsf{R}^h_k\rangle\pi(h+1) = \langle T, \varepsilon\rangle$$

که این یعنی داریم:

$$\forall j : 1 \leq j \leq h \rightarrow \langle \underline{\rho}, \langle l_j, \rho_j\rangle\rangle \in \mathcal{S}^r[\![\mathsf{L}_j : \mathsf{B}_j]\!]$$

$$\Rightarrow \langle \underline{\rho}, \pi\rangle \in \mathcal{S}^r[\![\mathsf{R}_k \bullet (?:T)^*]\!] \Rightarrow \langle \underline{\rho}, \pi\rangle \in \mathsf{prefix}(\mathcal{S}^\mathsf{r}[\![\mathsf{R} \bullet (?:T)^*]\!])$$

$$\blacktriangleright\blacktriangleright\ h \geq l :$$

در این صورت داریم:

$$\mathcal{M}^t\langle \underline{\rho}, \mathsf{R}_\mathsf{k}\rangle\pi = \langle T, \mathsf{R}^l_k\rangle$$

که یعنی:

$$\forall j : 1 \leq j \leq l \rightarrow \langle \underline{\rho}, \langle l_j, \rho_j\rangle\rangle \in \mathcal{S}^r[\![\mathsf{L}_j : \mathsf{B}_j]\!]$$

$$\Rightarrow \langle \underline{\rho}, \pi\rangle \in \mathsf{prefix}(\mathcal{S}^r[\![\mathsf{R}_k]\!]) \Rightarrow \langle \underline{\rho}, \pi\rangle \in \mathsf{prefix}(\mathcal{S}^\mathsf{r}[\![\mathsf{R} \bullet (?:T)^*]\!])$$

پس در کل می‌توانیم، بگوییم

$$\langle \underline{\rho}, \pi\rangle \in \mathsf{prefix}(\mathcal{S}^\mathsf{r}[\![\mathsf{R} \bullet (?:T)^*]\!])$$

و اثبات قضیه تمام می‌شود. $\square$

حال به اثبات تمامیت می‌پردازیم.

قضیه ۱۶.۳. (قضیه تمامیت): اگر P یک برنامه، R یک عبارت منظم و $\underline{\rho}$ یک محیط اولیه باشند، آنگاه داریم:

$$\mathsf{P}, \underline{\rho} \models \mathsf{R} \Rightarrow \mathsf{P}, \underline{\rho} \models_r \mathsf{R}$$

اثبات. با برهان خلف این قضیه را ثابت می‌کنیم. شکل اثبات تا حدی شبیه به اثبات درستی است.

$$\{\underline{\rho}\}\times\mathcal{S}^*[\![\mathsf{P}]\!] \nsubseteq \mathcal{M}\langle\underline{\rho},\mathsf{R}\rangle\mathcal{S}^*[\![\mathsf{P}]\!] \Rightarrow \exists\pi : \langle\underline{\rho},\pi\rangle \in \{\rho\}\times\mathcal{S}^*[\![\mathsf{P}]\!]\wedge\langle\underline{\rho},\pi\rangle \notin \mathcal{M}\langle\underline{\rho},\mathsf{R}\rangle\mathcal{S}^*[\![\mathsf{P}]\!]$$

اگر فرض کنیم $\mathsf{dnf}(\mathsf{R}) = \mathsf{R_1} + \mathsf{R_2} + \ldots + \mathsf{R_n}$ و علاوه بر این، با توجه به آنچه در اثبات درستی گفتیم، فرض کنیم:

$$\mathsf{R}_i \Leftrightarrow \mathsf{L_i^1} : \mathsf{B_i^1} \bullet \mathsf{L_i^2} : \mathsf{B_i^2} \bullet \ldots \bullet \mathsf{L_i^n} : \mathsf{B_i^n}$$

و

$$\pi = \langle l_1, \rho_1\rangle\langle l_2, \rho_2\rangle\ldots\langle l_l, \rho_l\rangle$$

می‌توانیم، در ادامه‌ی فرض خلف، نتیجه بگیریم:

$$\forall i : 1 \leq i \leq n \rightarrow \langle\underline{\rho},\pi\rangle \notin \mathcal{M}^\dagger\langle\underline{\rho},\mathsf{R}_i\rangle\mathcal{S}^*[\![\mathsf{P}]\!]$$

$$\Rightarrow \forall i : 1 \leq i \leq n \rightarrow \exists \mathsf{R}'_i :\Rightarrow \mathcal{M}^t\langle\underline{\rho},\mathsf{R}_i\rangle\pi = \langle F, \mathsf{R}_i^k\rangle$$

در این صورت، خواهیم داشت:

$$\exists j : \langle\underline{\rho},\langle l_j,\rho_j\rangle\rangle \notin \mathcal{S}^r[\![\mathsf{L_i^j} : \mathsf{B_i^j}]\!] \Rightarrow \langle\underline{\rho},\pi\rangle \notin \mathcal{S}^r[\![\mathsf{R}_i \bullet (?:T)^*]\!]$$

از نتیجه‌ی آخر می‌توانیم ثابت کنیم $\langle\underline{\rho},\pi\rangle \notin \mathsf{prefix}(\mathcal{S}^r[\![\mathsf{R}_i \bullet (?:T)^*]\!])$. چون اگر غیر از این باشد، یعنی اگر فرض کنیم $\langle\underline{\rho},\pi'\rangle$ عضو $\mathsf{prefix}(\mathcal{S}^r[\![\mathsf{R}_i \bullet (?:T)^*]\!])$ هست، اما عضو $\mathcal{S}^r[\![\mathsf{R}_i \bullet (?:T)^*]\!]$ نیست، در آن صورت، اگر طول π بزرگتر یا مساوی j باشد، به خاطر وجود $\langle l_j,\rho_j\rangle$ در π، خواهیم داشت $\pi \neq \pi'$ و اگر طول π' کمتر از j باشد، چون طول π قطعا بزرگتر یا مساوی j است(نتیجه از گزاره‌ی بالایی که دارای سور وجودی است)، پس باز هم $\pi \neq \pi'$. توجه شود که $\langle\underline{\rho},\pi\rangle \notin \mathcal{S}^r[\![\mathsf{R}_i \bullet (?:T)^*]\!]$ با توجه به آنچه گفتیم، با فرض در تناقض است و حکم ثابت می‌شود. $\square$

فصل ۴

وارسی مدل ساختارمند

در این فصل، به ادامه‌ی ساختارمندتر کردن کار می‌پردازیم. در فصل گذشته، ساختار عبارات منظم را به تعریف وارسی مدل اضافه کردیم و حالا می‌خواهیم، ساختار زبانمان را به کار اضافه کنیم. این آخرین تلاش [۶] برای گسترش کار بوده است. یعنی وارسی مدل در صورت جدید تعریف شده است و معادل بودن آن با صورت قبلی وارسی مدل ثابت شده است و پس از آن کار پایان می‌پذیرد.

چون تعریف صورت جدید روی ساختار زبان انجام گرفته است، جزئیات بسیار طولانی‌ای دارد. همین موضوع باعث شده است، تا اثبات برابری این صورت با صورت قبلی هم بسیار مفصل و حجیم باشد. این اثبات در [۶] به طور کامل حین معرفی هر حالت تعریف بیان شده است. بنابراین، از ارائه‌ی دوباره‌ی این جزئیات خودداری کرده‌ایم.

تعریف ۱.۴. تابع $\hat{\mathcal{M}}$ را از نوع $(\underline{\mathbb{EV}} \times \mathbb{R}) \rightarrow P(\mathfrak{S}^{+\infty}) \rightarrow (\underline{\mathbb{EV}} \times P(\mathfrak{S}^{+\infty}))$ وارسی مدل ساختارمند[1] می‌نامیم(ضابطه‌ی تابع در ادامه‌ی متن آمده است).

در ادامه، ممکن است به‌جای $\hat{\mathcal{M}}\langle\underline{\rho}, \mathsf{R}\rangle\mathcal{S}^*[\![\mathsf{P}]\!]$ از $\hat{\mathcal{M}}\langle\underline{\rho}, \mathsf{R}\rangle[\![\mathsf{P}]\!]$ استفاده کرده باشیم، یعنی در اشاره به تابع $\mathcal{S}^*$ به براکت‌ها $[\![\ ]\!]$ قناعت کرده باشیم.

تعریف روی ساختار مجموعه‌ی $\mathbb{S} \cup \mathbb{S}\mathbb{l} \cup \mathbb{P}$ انجام شده است. تقریبا کاری شبیه به اثبات لمی که در بحث تصمیم ناپذیری در فصل سوم داشتیم. در ادامه، قسمت‌های مختلف تعریف $\hat{\mathcal{M}}$ را به ازای برنامه‌ی P، محیط اولیه‌ی $\underline{\rho}$ و عبارت منظم Rتعریف می‌کنیم. یعنی در حال تعریف

[1] Structural Model Checking

$\hat{\mathcal{M}}\langle\underline{\rho}, \mathsf{R}\rangle\mathcal{S}^*[\![\mathsf{P}]\!]$ هستیم، روی ساختار برنامه‌ها یعنی P.

$$\blacktriangleleft\ \mathsf{P} = \mathsf{SI}:$$

$$\hat{\mathcal{M}}\langle\underline{\rho}, \mathsf{R}\rangle\mathcal{S}^*[\![\mathsf{P}]\!] = \bigcup_{i=1}^{n}\{\langle\underline{\rho}, \pi\rangle | \exists \mathsf{R}' \in \mathbb{R},\ \langle\pi, \mathsf{R}'\rangle\hat{\mathcal{M}}^{\dagger}\langle\underline{\rho}, \mathsf{R}_i\rangle\mathcal{S}^*[\![\mathsf{SI}]\!]\}$$
$$\text{where } \mathsf{dnf}(\mathsf{R}) = \mathsf{R}_1 + \mathsf{R}_2 + \ldots + \mathsf{R}_n$$

اثبات برابری این قسمت از تابع با صورت فصل قبل با اینکه $\hat{\mathcal{M}}^{\dagger}\langle\underline{\rho}, \mathsf{R}\rangle[\![\mathsf{SI}]\!]$ هنوز تعریف نشده است، در [۶] آمده است. استدلال این است که برابری $\hat{\mathcal{M}}\langle\underline{\rho}, \mathsf{R}\rangle[\![\mathsf{SI}]\!] = \mathcal{M}\langle\underline{\rho}, \mathsf{R}\rangle[\![\mathsf{SI}]\!]$ در فرض استقرا آمده است. کلیت اثبات هم این است که از باز کردن تعریف $\mathcal{M}\langle\underline{\rho}, \mathsf{R}\rangle[\![\mathsf{P}]\!]$ با استفاده‌ی مستقیم تعاریف و بدون تکنیک خاصی به $\hat{\mathcal{M}}\langle\underline{\rho}, \mathsf{R}\rangle[\![\mathsf{P}]\!]$ رسیده است.

در ادامه با توجه به تعریف قبل، به بیان تعریف $\hat{\mathcal{M}}^{\dagger}$ پرداخته شده است. این تنها بخش تابع $\hat{\mathcal{M}}$ است که معرفی نشده است و با مشخص شدن آن معنای $\hat{\mathcal{M}}$ به ازای برنامه‌های مختلف مشخص می‌شود.

این نکته را در نظر داریم که $\hat{\mathcal{M}}^{\dagger}$ در عمل روی مجموعه‌ی معنای برنامه‌ها تعریف می‌شود. مثلا، به ازای $\Pi \subseteq \mathfrak{S}^{+\infty}$ دلخواه که مساوی معنای یک برنامه نباشد، اینکه این تابع با یک محیط اولیه و یک عبارت منظم چه خروجی‌ای دارد، برای ما اهمیتی ندارد. در واقع، تعریف تابع اصلا به ازای چنین ورودی‌ای خروجی ندارد. به عبارت دیگر، تابع جزئی است. مشابه $\mathcal{M}^{\dagger}$ خروجی $\hat{\mathcal{M}}^{\dagger}$ هم یک زوج مرتب شامل π ای است که R را ارضا کرده است، به همراه یک عبارت منظم بدون + که بخشی از R را نشان می‌دهد که با π تطابق داده نشده است.

$$\blacktriangleleft\ \hat{\mathcal{M}}^{\dagger}\langle\underline{\rho}, \varepsilon\rangle[\![\mathsf{S}]\!] = \{\langle\pi, \varepsilon\rangle | \pi \in \mathcal{S}^*[\![\mathsf{S}]\!]\}$$

برای $\mathsf{SI} = \mathsf{SL}'\ \mathsf{S}$ و $\mathsf{R} \in \mathbb{R}^{\dagger} \cap \mathbb{R}^{+}$ داریم:

$$\blacktriangleleft\ \hat{\mathcal{M}}^{\dagger}\langle\underline{\rho}, \mathsf{R}\rangle[\![\mathsf{SI}]\!] = \hat{\mathcal{M}}^{\dagger}\langle\underline{\rho}, \mathsf{R}\rangle[\![\mathsf{SI}']\!]\cup$$
$$\{\langle\pi\langle \mathsf{at}[\![\mathsf{S}]\!], \rho\rangle\pi', \mathsf{R}''\rangle | \langle\pi\langle at[\![\mathsf{S}]\!], \rho\rangle, \mathsf{R}'\rangle \in \hat{\mathcal{M}}^{\dagger}\langle\underline{\rho}, \mathsf{R}\rangle[\![\mathsf{SI}']\!]\wedge$$
$$\langle\langle at[\![\mathsf{S}]\!], \rho\rangle\pi', \mathsf{R}''\rangle \in \hat{\mathcal{M}}^{\dagger}\langle\underline{\rho}, \mathsf{R}'\rangle[\![\mathsf{S}]\!]\}$$

از اینجا به بعد، با تعاریف طویل‌تری از آنچه تا حالا داشتیم، روبرو هستیم. هرچند که مفهوم چندان پیچیده‌ای پشت این تعاریف نیست. به طور خلاصه، تعریف بالا می‌گوید، تعریف $\hat{\mathcal{M}}^{\dagger}\langle\underline{\rho}, \mathsf{R}\rangle[\![\mathsf{SI}]\!]$

وابسته به تعریف $\hat{\mathcal{M}}^{\dagger}\langle\underline{\rho},\mathsf{R}\rangle[\![\mathsf{S}]\!]$ و $\hat{\mathcal{M}}^{\dagger}\langle\underline{\rho},\mathsf{R}\rangle[\![\mathsf{Sl}']\!]$ است. ردهای پیشوندی‌ای که داخل این دو مجموعه هستند، به یکدیگر چسبانده می‌شوند، به طوریکه اول ردهای داخل $\hat{\mathcal{M}}^{\dagger}\langle\underline{\rho},\mathsf{R}\rangle[\![\mathsf{Sl}']\!]$ قرار می‌گیرند و بعد ردهای داخل $\hat{\mathcal{M}}^{\dagger}\langle\underline{\rho},\mathsf{R}\rangle[\![\mathsf{S}]\!]$. ردهای داخل $\hat{\mathcal{M}}^{\dagger}\langle\underline{\rho},\mathsf{R}\rangle[\![\mathsf{Sl}']\!]$ به تنهایی نیز داخل $\hat{\mathcal{M}}^{\dagger}\langle\underline{\rho},\mathsf{R}\rangle[\![\mathsf{Sl}]\!]$ قرار می‌گیرند. این تعریف بر اساس تعریف $\mathcal{S}^{*}[\![\mathsf{Sl}]\!]$ ارائه شده است. اساس گرفتن تعریف تابع $\mathcal{S}^{*}$ در کنار توجه به تعریف تابع $\mathcal{M}$ که در فصل پیش ارائه شد، در ادامه‌ی تعریف $\hat{\mathcal{M}}^{\dagger}$ نیز حضور دارد.

برای $\mathsf{Sl}=\epsilon$ و $\mathsf{R}\in\mathbb{R}^{\dagger}\cap\mathbb{R}^{+}$ داریم:

$$\blacktriangleleft\ \hat{\mathcal{M}}^{\dagger}\langle\underline{\rho},\mathsf{R}\rangle[\![\mathsf{Sl}]\!] =$$

$$\{\langle\langle at[\![\mathsf{Sl}]\!],\rho\rangle,\mathsf{R}'\rangle|\langle\underline{\rho},\langle at[\![\mathsf{Sl}]\!],\rho\rangle\rangle\in\mathcal{S}^{r}[\![\mathsf{L}:\mathsf{B}]\!]\}$$

$$\text{where } \mathsf{fstnxt}(\mathsf{R})=\langle\mathsf{L}:\mathsf{B},\mathsf{R}'\rangle$$

یعنی خروجی تابع به ازای این ورودی مجموعه‌ای است، شامل همه‌ی ردهای پیشوندی تک عضوی‌ای که محیط آن‌ها اولین سر عبارت منظم($\mathsf{L}:\mathsf{B}$) را ارضا می‌کند. به عبارت دیگر، هر محیطی که این اتم را ارضا کند، برچسب این لیست عبارت‌های دستوری را در این مجموعه می‌آورد(به همراه ادامه‌ی عبارت منظم).

برای $\mathsf{S}=\ \mathsf{x}\doteq\mathsf{A};$ و $\mathsf{R}\in\mathbb{R}^{\dagger}\cap\mathbb{R}^{+}$ داریم:

$$\blacktriangleleft\ \hat{\mathcal{M}}^{\dagger}\langle\underline{\rho},\mathsf{R}\rangle[\![\mathsf{S}]\!] =$$

$$\{\langle\langle at[\![\mathsf{S}]\!],\rho\rangle,\mathsf{R}'\rangle|\langle\underline{\rho},\langle at[\![\mathsf{S}]\!],\rho\rangle\rangle\in\mathcal{S}^{r}[\![\mathsf{L}:\mathsf{B}]\!]\}$$

$$\cup\{\langle\langle at[\![\mathsf{S}]\!],\rho\rangle\langle aft[\![\mathsf{S}]\!],\rho[\mathsf{x}\leftarrow\mathcal{A}[\![\mathsf{A}]\!]\rho]\rangle,\varepsilon\rangle|\mathsf{R}'\in\mathbb{R}_{\varepsilon}\wedge$$

$$\langle\underline{\rho},\langle at[\![\mathsf{S}]\!],\rho\rangle\rangle\in\mathcal{S}^{r}[\![\mathsf{L}:\mathsf{B}]\!]\}$$

$$\cup\{\langle\langle at[\![\mathsf{S}]\!],\rho\rangle\langle aft[\![\mathsf{S}]\!],\rho[\mathsf{x}\leftarrow[\![\mathsf{A}]\!]\rho]\rangle,\mathsf{R}''\rangle|\mathsf{R}'\notin\mathbb{R}_{\varepsilon}\wedge$$

$$\langle\underline{\rho},\langle at[\![\mathsf{S}]\!],\rho\rangle\rangle\in\mathcal{S}^{r}[\![\mathsf{L}:\mathsf{B}]\!]\wedge\langle\mathsf{L}':\mathsf{B}',\mathsf{R}''\rangle=\mathsf{fstnxt}(\mathsf{R}')\wedge$$

$$\langle\underline{\rho},\langle aft[\![\mathsf{S}]\!],\rho[\mathsf{x}\leftarrow\mathcal{A}[\![\mathsf{A}]\!]\rho]\rangle\rangle\in\mathcal{S}^{r}[\![\mathsf{L}':\mathsf{B}']\!]\}$$

$$\text{where } \mathsf{fstnxt}(\mathsf{R})=\langle\mathsf{L}:\mathsf{B},\mathsf{R}'\rangle$$

این تابع عبارت دستوری را به همراه زوج مرتبی شامل محیط اولیه و عبارت منظم می‌گیرد، همان خروجی‌ای که در حالت قبلی برمی‌گرداند را برمی‌گرداند، سپس نسبت به اینکه پس از تغییر در محیط‌ها (در اثر اجرای عبارت دستوری مقدار دهی) یک رد پیشوندی با ادامه‌ی عبارت منظم سازگار باشد یا نباشد، زوج‌هایی را متشکل از رد پیشوندی و عبارت منظم به خروجی اضافه می‌کند.

از این ۴ حالت تنها اثبات حالت آخر در [۶] آورده شده است. اثبات دیگر حالات را هم می‌توان در همین اثبات که مفصل‌تر است، دید. اثبات سر راست است و در آن از جایگذاری تساوی‌های واضح استفاده شده است و جزئیات کافی دارد.

برای عبارت منظم $\mathsf{R} \in \mathbb{R}^{\dagger} \cup \mathbb{R}^{+}$ و $\mathsf{S} = \text{ if } (\mathsf{B})\ \mathsf{S_t}$ داریم:

$$\blacktriangleleft\ \hat{\mathcal{M}}^{\dagger}\langle \underline{\rho}, \mathsf{R}\rangle[\![\mathsf{S}]\!] =$$

$$\{\langle\langle at[\![\mathsf{S}]\!], \rho\rangle, \mathsf{R}'\rangle | \langle \underline{\rho}, \langle at[\![\mathsf{S}]\!], \rho\rangle\rangle \in \mathcal{S}^r[\![\mathsf{L}' : \mathsf{B}']\!]\}$$

$$\cup\{\langle\langle at[\![\mathsf{S}]\!], \rho\rangle\langle at[\![\mathsf{S_t}]\!], \rho\rangle\pi, \mathsf{R}'' | \mathcal{B}[\![\mathsf{B}]\!]\rho = T\wedge$$

$$\langle \underline{\rho}, \langle at[\![\mathsf{S}]\!], \rho\rangle\rangle \in \mathcal{S}^r[\![\mathsf{L}' : \mathsf{B}']\!]\wedge$$

$$\langle\langle at[\![\mathsf{S_t}]\!], \rho\rangle\pi, \mathsf{R}''\rangle \in \hat{\mathcal{M}}^{\dagger}\langle \underline{\rho}, \mathsf{R}'\rangle[\![\mathsf{S_t}]\!]\}$$

$$\cup\{\langle\langle at[\![\mathsf{S}]\!], \rho\rangle\langle aft[\![\mathsf{S}]\!], \rho\rangle, \varepsilon\rangle | \mathcal{B}[\![\mathsf{B}]\!]\rho = F \wedge \mathsf{R}' \in \mathbb{R}_{\varepsilon}\wedge$$

$$\langle \underline{\rho}, \langle[\![\mathsf{S}]\!], \rho\rangle\rangle \in \mathcal{S}^r[\![\mathsf{L}' : \mathsf{B}']\!]\}$$

$$\cup\{\langle\langle at[\![\mathsf{S}]\!], \rho\rangle\langle aft[\![\mathsf{S}]\!], \rho\rangle, \mathsf{R}''\rangle | \mathcal{B}[\![\mathsf{B}]\!]\rho = F \wedge \mathsf{R}' \notin \mathbb{R}_{\varepsilon}\wedge$$

$$\langle \underline{\rho}, \langle at[\![\mathsf{S}]\!], \rho\rangle\rangle \in \mathcal{S}^r[\![\mathsf{L}' : \mathsf{B}']\!] \wedge \langle \mathsf{L}'' : \mathsf{B}'', \mathsf{R}''\rangle = \mathsf{fstnxt}(\mathsf{R}')\wedge$$

$$\langle \underline{\rho}, \langle aft[\![\mathsf{S}]\!], \rho\rangle\rangle \in \mathcal{S}^r[\![\mathsf{L}'' : \mathsf{B}'']\!]\}$$

$$\text{where } \mathsf{fstnxt}(\mathsf{R}) = \langle \mathsf{L}' : \mathsf{B}', \mathsf{R}'\rangle$$

برای عبارت منظم $\mathsf{R} \in \mathbb{R}^{\dagger} \cup \mathbb{R}^{+}$ و $\mathsf{S} = \text{ if } (\mathsf{B})\ \mathsf{S_t} \text{ else } \mathsf{S_f}$ داریم:

$$\blacktriangleleft\ \hat{\mathcal{M}}^{\dagger}\langle \underline{\rho}, \mathsf{R}\rangle[\![\mathsf{S}]\!] =$$

$$\{\langle\langle at[\![\mathsf{S}]\!], \rho\rangle, \mathsf{R}'\rangle | \langle \underline{\rho}, \langle at[\![\mathsf{S}]\!], \rho\rangle\rangle \in \mathcal{S}^r[\![\mathsf{L}' : \mathsf{B}']\!]\}$$

$$\cup\{\langle\langle at[\![\mathsf{S}]\!], \rho\rangle\langle at[\![\mathsf{S_t}]\!], \rho\rangle\pi, \mathsf{R}'' | \mathcal{B}[\![\mathsf{B}]\!]\rho = T\wedge$$

$$\langle\underline{\rho},\langle at[\![\mathsf{S}]\!],\rho\rangle\rangle\in\mathcal{S}^r[\![\mathsf{L}':\mathsf{B}']\!]\wedge$$

$$\langle\langle at[\![\mathsf{S_t}]\!],\rho\rangle\pi,\mathsf{R}''\rangle\in\hat{\mathcal{M}}^{\dagger}\langle\underline{\rho},\mathsf{R}'\rangle[\![\mathsf{S_t}]\!]\}$$

$$\cup\{\langle\langle at[\![\mathsf{S}]\!],\rho\rangle\langle at[\![\mathsf{S_f}]\!],\rho\rangle\pi,\mathsf{R}''|\mathcal{B}[\![\mathsf{B}]\!]\rho=F\wedge$$

$$\langle\underline{\rho},\langle at[\![\mathsf{S}]\!],\rho\rangle\rangle\in\mathcal{S}^r[\![\mathsf{L}':\mathsf{B}']\!]\wedge$$

$$\langle\langle at[\![\mathsf{S_f}]\!],\rho\rangle\pi,\mathsf{R}''\rangle\in\hat{\mathcal{M}}^{\dagger}\langle\underline{\rho},\mathsf{R}'\rangle[\![\mathsf{S_f}]\!]\}$$

$$\text{where fstnxt}(\mathsf{R})=\langle\mathsf{L}':\mathsf{B}',\mathsf{R}'\rangle$$

دو قسمت بالا در مورد عبارت‌های دستوری شرطی هستند. در مورد نوع اولی شرطی، یک رد پیشوندی را در معنای S در نظر بگیرید. بسته به اینکه طبق محیط حاضر در اولین وضعیت این رد پیشوندی، عبارت بولی مقدار صحیح یا غلط داشته باشد، حضور این رد پیشوندی(در یک زوج، به همراه یک عبارت منظم) داخل $\hat{\mathcal{M}}^{\dagger}\langle\underline{\rho},\mathsf{R}\rangle[\![\mathsf{S}]\!]$ تعیین می‌شود. اگر عبارت بولی در محیط مذکور مقدار صحیح داشته باشد، در معنای نوع اول عبارت دستوری شرطی، پس از تطبیق سر عبارت منظم با اولین وضعیت هر رد پیشوندی، بر اساس اینکه در دومین وضعیت رد پیشوندی، عبارت بولی برقرار باشد یا نباشد، وابسته به این می‌شود که آیا اگر از وضعیت دوم به بعد این رد پیشوندی(که خود یک رد پیشوندی است) در $\hat{\mathcal{M}}^{\dagger}\langle\underline{\rho},\mathsf{R}'\rangle[\![\mathsf{S_t}]\!]$ (که R' نیز دم R است) حضور دارد یا خیر. اگر عبارت بولی در محیط مذکور ارزش غلط داشته باشد، حضور رد پیشوندی در $\hat{\mathcal{M}}^{\dagger}\langle\underline{\rho},\mathsf{R}\rangle[\![\mathsf{S}]\!]$ وابسته به سازگاری وضعیت دوم رد پیشوندی با سر R' خواهد بود.

در نوع دوم عبارت دستوری شرطی نیز، تعریف شبیه به نوع اول است، با این تفاوت که اگر عبارت بولی در محیط اولین وضعیت رد پیشوندی ارزش غلط داشته باشد، اتفاقی شبیه به حالت درست می‌افتد.

برای عبارت منظم $\mathsf{R}\in\mathbb{R}^{\dagger}\cup\mathbb{R}^{+}$ و $\mathsf{S}=\ \mathsf{break};$ داریم:

$$\blacktriangleleft\ \hat{\mathcal{M}}^{\dagger}\langle\underline{\rho},\mathsf{R}\rangle[\![\mathsf{S}]\!]=$$

$$\{\langle\langle at[\![\mathsf{S}]\!],\rho\rangle,\mathsf{R}'\rangle|\langle\underline{\rho},\langle at[\![\mathsf{S}]\!],\rho\rangle\rangle\in\mathcal{S}^r[\![\mathsf{L}:\mathsf{B}]\!]\}$$

$$\cup\{\langle\langle at[\![\mathsf{S}]\!],\rho\rangle\langle brk-to[\![\mathsf{S}]\!],\rho\rangle,\varepsilon\rangle|\mathsf{R}'\in\mathbb{R}_{\varepsilon}\wedge$$

$$\langle\underline{\rho},\langle at[\![\mathsf{S}]\!],\rho\rangle\rangle\in\mathcal{S}^r[\![\mathsf{L}:\mathsf{B}]\!]\}$$

$$\cup\{\langle\langle at[\![\mathsf{S}]\!],\rho\rangle\langle brk-to[\![\mathsf{S}]\!],\rho\rangle,\mathsf{R}''\rangle|\mathsf{R}'\notin\mathbb{R}_{\varepsilon}\wedge$$

$$\langle \underline{\rho}, \langle at[\![\mathsf{S}]\!], \rho\rangle\rangle \in \mathcal{S}^r[\![\mathsf{L} : \mathsf{B}]\!] \wedge \langle \mathsf{L}' : \mathsf{B}', \mathsf{R}''\rangle = \mathsf{fstnxt}(\mathsf{R}')\wedge$$

$$\langle \underline{\rho}, \langle brk - to[\![\mathsf{S}]\!], \rho\rangle\rangle \in \mathcal{S}^r[\![\mathsf{L}' : \mathsf{B}']\!]\}$$

$$\text{where } \mathsf{fstnxt}(\mathsf{R}) = \langle \mathsf{L}' : \mathsf{B}', \mathsf{R}'\rangle$$

با توجه به موارد قبلی، اینکه این قسمت از تعریف چه معنایی دارد و به چه علت به این شکل است، قابل درک است.

برای عبارت منظم $\mathsf{R} \in \mathbb{R}^{\dagger} \cup \mathbb{R}^{+}$ و : $\mathsf{S} =$ while (B) $\mathsf{S_b}$ داریم:

$$\blacktriangleleft\ \hat{\mathcal{M}}^{\dagger}\langle \underline{\rho}, \mathsf{R}\rangle[\![\mathsf{S}]\!] = lfp^{\subseteq}\ (\hat{\mathcal{F}}^{\dagger}\langle \underline{\rho}, \mathsf{R}\rangle[\![\mathsf{S}]\!])$$

$$\text{where } \hat{\mathcal{F}}^{\dagger}\langle \underline{\rho}, \mathsf{R}\rangle X = \{\langle\langle at[\![\mathsf{S}]\!], \rho\rangle, \mathsf{R}'\rangle | \rho \in \mathbb{EV} \wedge \langle \underline{\rho}, \langle at[\![\mathsf{S}]\!], \rho\rangle\rangle \in \mathcal{S}^r[\![\mathsf{L}' : \mathsf{B}']\!]\}$$

$$\cup\{\langle \pi_2\langle at[\![\mathsf{S}]\!], \rho\rangle\langle aft[\![\mathsf{S}]\!], \rho\rangle, \varepsilon\rangle | \langle \pi_2\langle at[\![\mathsf{S}]\!], \rho\rangle, \varepsilon\rangle \in X\wedge$$

$$\mathcal{B}[\![\mathsf{B}]\!]\rho = F\}$$

$$\cup\{\langle \pi_2\langle at[\![\mathsf{S}]\!], \rho\rangle\langle aft[\![\mathsf{S}]\!], \rho\rangle, \varepsilon\rangle | \langle \pi_2\langle at[\![\mathsf{S}]\!], \rho\rangle, \mathsf{R}''\rangle \in X\wedge$$

$$\mathcal{B}[\![\mathsf{B}]\!]\rho = F \wedge \mathsf{R}'' \notin \mathbb{R}_{\varepsilon} \wedge \langle \mathsf{L}' : \mathsf{B}', \mathsf{R}'\rangle = \mathsf{fstnxt}(\mathsf{R}'') \wedge \mathsf{R}' \in \mathbb{R}_{\varepsilon}\wedge$$

$$\langle \underline{\rho}, \langle at[\![\mathsf{S}]\!], \rho\rangle\rangle \in \mathcal{S}^r[\![\mathsf{L}' : \mathsf{B}']\!]\}$$

$$\cup\{\langle \pi_2\langle at[\![\mathsf{S}]\!], \rho\rangle\langle aft[\![\mathsf{S}]\!], \rho\rangle, \mathsf{R}'\rangle | \langle \pi_2\langle at[\![\mathsf{S}]\!], \rho\rangle, \mathsf{R}''\rangle \in X\wedge$$

$$\mathcal{B}[\![\mathsf{B}]\!]\rho = F \wedge \mathsf{R}'' \notin \mathbb{R}_{\varepsilon} \wedge \langle \mathsf{L}' : \mathsf{B}', \mathsf{R}'\rangle = \mathsf{fstnxt}(\mathsf{R}'') \wedge \mathsf{R}''' \notin \mathbb{R}_{\varepsilon}\wedge$$

$$\langle \underline{\rho}, \langle at[\![\mathsf{S}]\!], \rho\rangle\rangle \in \mathcal{S}^r[\![\mathsf{L}' : \mathsf{B}']\!] \wedge \langle \mathsf{L}'' : \mathsf{B}'', \mathsf{R}'\rangle = \mathsf{fstnxt}(\mathsf{R}''')\wedge$$

$$\langle \underline{\rho}, \langle aft[\![\mathsf{S}]\!], \rho\rangle\rangle \in \mathsf{S}^r[\![\mathsf{L}'' : \mathsf{B}'']\!]\}$$

$$\cup\{\langle \pi_2\langle at[\![\mathsf{S}]\!], \rho\rangle\langle at[\![\mathsf{S_b}]\!], \rho\rangle\pi_3, \varepsilon\rangle | \langle \pi_2\langle at[\![\mathsf{S}]\!], \rho\rangle, \varepsilon\rangle \in X\wedge$$

$$\mathcal{B}[\![\mathsf{B}]\!]\rho = T \wedge \langle at[\![\mathsf{S_b}]\!], \rho\rangle\pi_3 \in \mathcal{S}^*[\![\mathsf{S_b}]\!]\}$$

$$\cup\{\langle \pi_2\langle at[\![\mathsf{S}]\!], \rho\rangle\langle at[\![\mathsf{S_b}]\!], \rho\rangle\pi_3, \varepsilon\rangle | \langle \pi_2\langle at[\![\mathsf{S}]\!], \rho\rangle, \mathsf{R}''\rangle \in X\wedge$$

$$\mathcal{B}[\![\mathsf{B}]\!]\rho = T \wedge \mathsf{R}'' \notin \mathbb{R}_{\varepsilon} \wedge \langle \mathsf{L} : \mathsf{B}, \varepsilon\rangle = \mathsf{fstnxt}(\mathsf{R}'')\wedge$$

$$\langle \underline{\rho}, \langle at[\![\mathsf{S}]\!], \rho\rangle\rangle \in \mathcal{S}^r[\![\mathsf{L} : \mathsf{B}]\!] \wedge \langle at[\![\mathsf{S_b}]\!], \rho\rangle\pi_3 \in \mathcal{S}^*[\![\mathsf{S_b}]\!]\}$$

$$\cup\{\langle \pi_2\langle at[\![\mathsf{S}]\!],\rho\rangle\langle at[\![\mathsf{S_b}]\!],\rho\rangle\pi_3,\mathsf{R}'\rangle|\langle \pi_2\langle at[\![\mathsf{S}]\!],\rho\rangle,\mathsf{R}''\rangle\in X\wedge$$
$$\mathcal{B}[\![\mathsf{B}]\!]\rho = T\wedge \mathsf{R}''\notin \mathbb{R}_\varepsilon \wedge \langle \mathsf{L}:\mathsf{B},\mathsf{R}''''\rangle = \mathsf{fstnxt}(\mathsf{R}'')\wedge$$
$$\langle \underline{\rho},\langle at[\![\mathsf{S}]\!],\rho\rangle\rangle \in \mathcal{S}^r[\![\mathsf{L}:\mathsf{B}]\!]\wedge \mathsf{R}''''\notin\mathbb{R}_\varepsilon\wedge$$
$$\langle \mathsf{L}':\mathsf{B}',\mathsf{R}'''\rangle = \mathsf{fstnxt}(\mathsf{R}'''')\wedge\langle \underline{\rho},\langle at[\![\mathsf{S_b}]\!],\rho\rangle\rangle\in\mathcal{S}^r[\![\mathsf{L}':\mathsf{B}']\!]\wedge$$
$$\langle\langle at[\![\mathsf{S_b}]\!],\rho\rangle\pi_3,\mathsf{R}'\rangle\in\hat{\mathcal{M}}^\dagger\langle\underline{\rho},\mathsf{R}'''\rangle[\![\mathsf{S_b}]\!]\}$$
$$\text{where } \langle \mathsf{L}':\mathsf{B}',\mathsf{R}'\rangle = \mathsf{fstnxt}(\mathsf{R})$$

تفاوت تعریف $\hat{\mathcal{M}}^\dagger$ برای عبارت دستوری حلقه با سایر عبارت‌های دستوری، حضور یک تابع به نام $\hat{\mathcal{F}}^\dagger$ در تعریف معنای آن است. در واقع، وارسی مدل به صورت کوچک‌ترین نقطه‌ثابت این تابع تعریف می‌شود. این همان کاری است که در تعریف معنای اجزای زبان هم انجام شد و چون می‌خواهیم ساختار زبان را به صورت وارسی مدل اضافه کنیم، انتظار داریم که عملگر نقطه‌ثابت هم در تعریف حضور پیدا کند. تابع $\hat{\mathcal{F}}^\dagger$ مثل یک دور اجرای حلقه عمل می‌کند، منتها در همین حین، سازگاری ردهای پیشوندی را با عبارت منظم بررسی می‌کند و ردهای پیشوندی‌ای را که با عبارت منظم ناسازگار هستند، مجموعه‌ی خروجی‌اش حذف می‌کند.

برای عبارت منظم $\mathsf{R}\in\mathbb{R}^\dagger\cup\mathbb{R}^+$ و $\mathsf{S} = ;$: داریم:

$$\blacktriangleleft\ \hat{\mathcal{M}}^\dagger\langle\underline{\rho},\mathsf{R}\rangle[\![\mathsf{S}]\!] = \{\langle\langle at[\![\mathsf{S}]\!],\rho\rangle,\mathsf{R}'\rangle|\langle\underline{\rho},\langle at[\![\mathsf{S}]\!],\rho\rangle\rangle\in\mathcal{S}^r[\![\mathsf{L}:\mathsf{B}]\!]\}\ \blacktriangleright$$
$$\text{where } \mathsf{fstnxt}(\mathsf{R}) = \langle \mathsf{L}:\mathsf{B},\mathsf{R}'\rangle$$

برای عبارت منظم $\mathsf{R}\in\mathbb{R}^\dagger\cup\mathbb{R}^+$ و $\mathsf{S} = \{\mathsf{Sl}\}$: داریم:

$$\blacktriangleleft\ \hat{\mathcal{M}}^\dagger\langle\underline{\rho},\mathsf{R}\rangle[\![\{\mathsf{Sl}\}]\!] = \hat{\mathcal{M}}^\dagger\langle\underline{\rho},\mathsf{R}\rangle[\![\mathsf{Sl}]\!]\ \blacktriangleright$$

همین طور، درست بودن یک ویژگی $\mathsf{R}\in\mathbb{R}$ را برای برنامه‌ی P و محیط اولیه‌ی $\underline{\rho}$ با

$$\mathsf{P},\underline{\rho}\models_s\mathsf{R}$$

نشان می‌دهیم و برقرار بودن این شرط به شکل زیر تعریف می‌شود:

$$\mathsf{P},\underline{\rho}\models_s\mathsf{R}\iff\{\underline{\rho}\}\times\mathcal{S}^*[\![\mathsf{P}]\!]\subseteq\hat{\mathcal{M}}\langle\underline{\rho},\mathsf{R}\rangle\mathcal{S}^*[\![\mathsf{P}]\!]$$

در اینجا تعریف توابع مربوط به وارسی مدل ساختارمند به پایان می‌رسد.

نتیجه گیری

دیدیم که صوری‌گری روش وارسی مدل در ادبیات نظریه‌ی تعبیر مجرد به چه شکل است و در همین حال سعی کردیم این صوری‌گری را شفاف‌تر و واضح‌تر از [۶] بیان کنیم. همین طور دیدیم که می‌توان به جای منطق‌های زمانی از عبارات منظم در روش وارسی مدل استفاده کرد.

همان طور که در [۶] آمده است و در فصل دوم به‌طور واضح‌تری نشان دادیم، این روش قابل پیاده‌سازی نیست. در [۶] نزدیک کردن کار به پیاده‌سازی را از طریق متناهی کردن مجموعه‌ی وضعیت‌ها ممکن می‌داند. به هر حال، در واقعیت هم مجموعه‌ی وضعیت‌ها متناهی است، چون حافظه‌ها متناهی هستند و این فرض می‌تواند صوری‌گری را یک قدم به واقعیت نزدیک‌تر کند.

ایده‌ای که ما برای نزدیک کردن این کار به پیاده‌سازی داریم این است که عبارات منظم محدودتر شود. اگر علاوه‌بر متناهی کردن مجموعه‌ی وضعیت‌ها، دو عملگر * و $^+$ را از عبارات منظم حذف کنیم، صوری‌گری احتمالا قابل پیاده‌سازی خواهد شد. درست است که حذف این دو عملگر از قدرت بیان ویژگی‌ها کم می‌کند، اما شاید در عمل، همان قدرت بیان باقی مانده برای بیان بسیاری از ویژگی‌ها کافی باشد.

واژه‌نامهٔ فارسی به انگلیسی

ا

اتم	Atom
اجتماع	Union
اجرا	Execution
ادات شفر	Sheffer Operator
ادات کلاسیک	Classical Connective
ادات وجهی	Modal Connective
ارضا کردن	Satisfy
ارضاپذیر	Satisfiable
استقرا	Induction
استقرایی	Inductive
اصل	Law
اعمال	Apply
افراز	Partition
الگوریتم	Algorithm
الگوریتم زمانبندی	Scheduling Algorithm
انتخاب	Choice

ب

بازتابی	Reflexive
بازگشتی	Recursive
برچسب	Label
برنامه کامپیوتری	Computer Program
بستار پیشوندی	Prefix Closure

پ

پایتون	Python
پخش پذیری	Distributivity
پردازنده	Processor
پیاده‌سازی	Implementation

ت

تابع	Function
تابع جزئی	Partial Function
تصمیم پذیری	Decidability
تصمیم ناپذیر	Undecidable
تعبیر مجرد	Abstract Interpretation
تقارنی	Symmetric
تمامیت	Completeness
توقف	Halt
توقف پذیر	Terminable
توقف پذیری	Halting
تهی	Empty

ج

چ

ح

خ

د

ر

رابطه .... Relation
رد پیشوندی .... Prefix Trace
روش صوری .... Formal Method

ز

زبان برنامه نویسی دستوری .... Imperative Programming Langugae
زبان طبیعی .... Natural Language
زوج مرتب .... Pair
زیرمجموعه .... Subset

س

ساختار .... Structure
ساختار جبری .... Algebraic Structure
سازگاری .... Consistency
سر و دم .... First Next

ش

شرط .... Conditional
شرطی .... Arrow
شرکت پذیر .... Associative
شمارای نامتناهی .... Countably Infinite

ص

صورت .... Form
صوری .... Formal
صوری‌گری .... Formalization

ع

عبارات منظم .... Regular Expressions
عبارات منظم بدون انتخاب .... Choice-free Regular Expressions
عبارات منظم تهی .... Empty Regular Expressions
عبارات منظم ناتهی .... Non-empty Regular Expressions
عبارت .... Expression
عبارت بولی .... Boolean Expression
عبارت حسابی .... Arithmetic Expression
عبارت دستوری .... Statement
عدد صحیح مثبت .... Positive Integers
عدد صحیح منفی .... Negative Integer
عدد طبیعی .... Natural Number
عضو .... Member
عملگر .... Operator
عملگر بولی .... Boolean Operator
عملگر دوتایی .... Binary Operator
عملگر ضرب .... Multiplication Operator
عملگر یگانی .... Unary Operator

غ

غلط .... False

غیر قطعی .... Non-deterministic
غیر هم‌ارز .... Unequivalent

ف

فرازبان .... Metalanguage
فرض استقرا .... Induction Hyposesis
فرم نرمال فصلی .... Disjunctive Normal Form
فرمول .... Formula
فرمول اتمی .... Atomic Formula
فصل .... Disjunction

ق

قابل بیان .... Expressible
قانون جذب .... Absorption Law
قدرت بیان .... Expressivity
قضیه‌ی نستر_تارسکی .... Knaster-Tarski Theorem

ک

کوچکترین نقطه‌ثابت .... Least Fixpoint

گ

گردایه .... Collection
گردش کار .... Workflow

ل

لیست عبارت‌های دستوری .... Statement List

م

ماشین تورینگ .... Turing Machine

ماشین حالات متناهی .... Finite State Machine

ماشین رجیستر .... Register Machine

ماکسیمال .... Maximal

متعدی .... Transitive

متغیر .... Variable

متناهی .... Finite

مجموعه .... Set

محیط .... Environment

محیط اولیه .... Initial Environment

مدل .... Model

مدل محاسبه .... Model of Computation

معتبر .... Valid

معماری .... Architecture

معناشناسی .... Semantics

معناشناسی رد پیشوندی .... Prefix Trace Semantics

معناشناسی صوری .... Formal Semantics

معناشناسی عبارات منظم .... Regular Expressions Semantics

مقدار .... Value

مقدار بولی .... Boolean Value

منطق .... Logic

منطق زمانی .... Temporal Logic

منطق زمانی خطی .... Linear Temporal Logic

ن

و

ه

ی

واژه‌نامهٔ انگلیسی به فارسی

A

Absorption Law .... قانون جذب
Abstract Interpretation .... تعبیر مجرد
Algebraic Structure .... ساختار جبری
Algorithm .... الگوریتم
Algorithm Theory .... نظریه الگوریتم
Apply .... اعمال
Architecture .... معماری
Arithmetic Expression .... عبارت حسابی
Arrow .... شرطی
Associative .... شرکت پذیر
Atom .... اتم
Atomic Formula .... فرمول اتمی

B

Binary Operator .... عملگر دوتایی
Boolean Expression .... عبارت بولی
Boolean Operator .... عملگر بولی
Boolean Value .... مقدار بولی

C

Case .... حالت
Choice .... انتخاب
Choice-free Regular Expressions .... عبارات منظم بدون انتخاب
Class .... دسته
Classical Connective .... ادات کلاسیک
Classical Logic .... منطق کلاسیک
Classical Propositional Logic .... منطق گزاره‌ای کلاسیک
Collection .... گردایه
Completeness .... تمامیت
Computational Complexity Theory .... نظریه پیچیدگی محاسبه
Computer Program .... برنامه کامپیوتری
Concatenation .... چسباندن
Conditional .... شرط
Consistency .... سازگاری
Countably Infinite .... شمارای نامتناهی

D

Decidability .... تصمیم پذیری
Disjunction .... فصل
Disjunctive Normal Form .... فرم نرمال فصلی
Distributivity .... پخش پذیری
Domain .... دامنه

E

Empty .... تهی

Empty Regular Expressions .... عبارات منظم تهی
Environment .... محیط
Equivalence .... هم‌ارزی
Equivalent .... هم‌ارز
Execution .... اجرا
Expressible .... قابل بیان
Expression .... عبارت
Expressivity .... قدرت بیان

F

False .... غلط
Finite .... متناهی
Finite State Machine .... ماشین حالات متناهی
Fixpoint .... نقطه‌ثابت
Form .... صورت
Formal .... صوری
Formal Method .... روش صوری
Formal Semantics .... معناشناسی صوری
Formalization .... صوری‌گری
Formula .... فرمول
First Next .... سر و دم
Function .... تابع

H

Halt .... توقف
Halting .... توقف پذیری

I

J

K

L

M

Maximal .... ماکسیمال
Member .... عضو
Memory .... حافظه
Metalanguage .... فرازبان
Modal Connective .... ادات وجهی
Modal Logic .... منطق موجهات
Model .... مدل
Model Checking .... وارسی مدل
Model of Computation .... مدل محاسبه
Monotonic .... یکنوا
Multiplication Operator .... عملگر ضرب

N

Natural Language .... زبان طبیعی
Natural Number .... عدد طبیعی
Negative Integer .... عدد صحیح منفی
Non-deterministic .... غیر قطعی
Non-empty Regular Expressions .... عبارات منظم ناتهی

O

Operator .... عملگر

P

Pair .... زوج مرتب

Partial Function .... تابع جزئی
Partition .... افراز
Positive Integers .... عدد صحیح مثبت
Prefix Closure .... بستار پیشوندی
Prefix Trace .... رد پیشوندی
Prefix Trace Semantics .... معناشناسی رد پیشوندی
Processor .... پردازنده
Property .... ویژگی
Python .... پایتون

R

Recursive .... بازگشتی
Reflexive .... بازتابی
Register Machine .... ماشین رجیستر
Regular Expressions .... عبارات منظم
Regular Expressions Semantics .... معناشناسی عبارات منظم
Regular Model Checking .... وارسی مدل منظم
Relation .... رابطه

S

Satisfiable .... ارضاپذیر
Satisfy .... ارضا کردن
Scheduling Algorithm .... الگوریتم زمانبندی
Semantics .... معناشناسی
Semantics of Regular Expressions .... معناشناسی عبارات منظم
Sequence .... دنباله
Set .... مجموعه

T

U

V

Valid .... معتبر
Value .... مقدار
Variable .... متغیر
Verification .... درستی‌یابی

W

Well-defined .... خوش تعریف
Workflow .... گردش کار

Bibliography

[1] Committee to review chinook zd 576 crash. report from the select committee on chinook zd 576., Feb 2002.

[2] A. S. E. Al. Mars climate orbiter mishap investigation boord phase i report., November 1999.

[3] A. Chlipala. *Certified Programming with Dependent Types: A Pragmatic Introduction to Coq Proof Assistant.* MIT Press, 2022.

[4] E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching-time temporal logic. In D. Kozen, editor, *Logic of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer, 1981.

[5] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model checking.* MIT Press, London, Cambridge, 1999.

[6] P. Cousot. Calculational design of a regular model checker by abstract interpretation. In R. M. Hierons and M. Mosbah, editors, *ICTAC*, volume 11884 of *Lecture Notes in Computer Science*, pages 3–21. Springer, 2019.

[7] P. Cousot. *Principals of Abstract Interpretation.* MIT Press, 2021.

[8] P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *POPL '77: Proceedings of the 4th ACM SIGACT-SIGPLAN symposium on Principles of programming languages*, pages 238–252. ACM Press, 1977.

[9] M. Davis and E. Weyuker. *Computability, Complexity, and Languages.* Academic Press, New York, 1983.

[10] D. Harel, D. Kozen, and J. Tiuryn. Dynamic logic. In *Handbook of philosophical logic*, pages 99–217. Springer, 2001.

[11] C. A. R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580, 1969.

[12] M. Huth and M. Ryan. *Logic in computer science : modelling and reasoning about systems.* Cambridge University Press, Cambridge [U.K.]; New York, 2004.

[13] R. M. John E. Hopcroft and J. D. Ullman. *Introduction to automata theory, languages, and computation.* Addison-Wesley, 2003.

[14] X. R. K. Yi. *Introduction to Static Analysis: An Abstract Interpretation Perspective.* MIT Press, 2020.

[15] S. Kleene. Representation of Events in Nerve Nets and Finite Automata. In C. Shannon and J. McCarthy, editors, *Automata Studies*, pages 3–41. Princeton University Press, 1956.

[16] D. Koze. On kleene algebras and closed semirings. *Springer Berlin Heidelberg*, 1990.

[17] S. A. Kripke. A completeness theorem in modal logic1. *The journal of symbolic logic*, 24(1):1–14, 1959.

[18] J. Lions. Ariane 5 Flight 501 Failure: Report of the Inquiry Board, July 1996.

[19] M. Mukund. Linear-time temporal logic and buchi automata. *Tutorial talk, Winter School on Logic and Computer Science, Indian Statistical Institute, Calcutta*, 1997.

[20] G. J. Myers, C. Sandler, and T. Badgett. *The art of software testing.* John Wiley & Sons, Hoboken and N.J, 3rd ed edition, 2012.

[21] B. C. Pierce, A. Azevedo de Amorimand Chris Casinghino, M. Gaboardi, M. Greenberg, C. Hriţcu, V. Sjöberg, A. Tolmach, and B. Yorgey. *Programming Language Foundations.* Software Foundations series, volume 2. Electronic textbook, May 2018.

[22] H. G. Rice. Classes of recursively enumerable sets and their decision problems. *Transactions of the American Mathematical Society*, 74(2):358–366, 1953.

[23] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific journal of Mathematics*, 5(2):285–309, 1955.

[24] G. Winskel. *The formal semantics of programming languages - an introduction.* Foundation of computing series. MIT Press, 1993.

Abstract

Model checking is a reliable method for program verification. Different forms of this method use temporal logic to express properties, which is not commonly accepted by programmers. In this thesis, it has been tried to represent a new form of model checking that has been stated in the literature of abstract interpretation theory and uses regular expressions for expressing program properties instead of modal logic.

After representing basic notions, three novel forms of model checking have been introduced. The first form has no structure and is merely expressed in a new literature; the second form has added the structure of regular expressions to itself; and the third form has the structure of programs in it to get closer to implementation. The equivalence of the three forms has been studied and discussed as well.

Kew Words: Model Checking, Abstract Interpretation, Denotational Semantics, Formal Verification, Static Analysis, Formal Verification of Computer Programs

College of Science
School of Mathematics, Statistics, and Computer Science

Improving Model Checking with Abstract Interpretation

Pouya Partow

Supervisor: Majid Alizadeh
Co-Supervisor: Mojtaba Mojtahedi

A thesis submitted to Graduate Studies Office
in partial fulfillment of the requirements for the degree of
Master of Science in
Computer Science

2023