



دانشکده‌گان علوم
دانشکده ریاضی، آمار و علوم کامپیوتر

بهبود روش واریسی مدل با استفاده از تعبیر مجرد

نگارنده

بویا 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 واری 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 یک بیان جدید از روش واری مدل مورد شرح و بررسی قرار گیرد که در آن به کمک نظریه تعبیر مجرد به جای منطق موجهات از عبارات منظم استفاده شده است.

پس از ارائه‌ی مفاهیم اولیه، به سه صورت متفاوت به بیانی جدید از روش واری مدل پرداخته‌ایم. صورت اول ساختار خاصی ندارد و صرفاً 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: واری مدل، نظریه تعبیر مجرد، معناشناسی دلالتی، درستی‌یابی صوری، تحلیل ایستا، درستی‌یابی برنامه‌های کامپیوتری

تقديم به

تقديم 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 نوشته‌شده 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 بوده‌اند، همین‌طور به‌سادگی می‌توان فجایع دیگری از این دست را در زندگی روزمره‌ی انسان‌ها متصور شد.

برای تعیین صحت کارکرد برنامه‌های کامپیوتری روش‌های متفاوتی ابداع شده‌اند که در ادامه به‌طور 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 اصلاً وجود ندارد. قضیه رایس (به‌طور غیر رسمی) بیان می‌کند که مسئله‌ی بررسی هر خاصیت غیر بدیهی، برای همه‌ی 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 برنامه 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 تست‌ها 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 گفته می‌شود. معروف‌ترین روش‌های ایستا؛ روش واریسی مدل، روش‌های استنتاجی و استفاده از نظریه تعبیر مجرد است.

در روش واریسی 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 صحبت کند، ویژگی‌های مورد بررسی را بیان می‌کنیم و در نهایت صحت ویژگی‌های بیان‌شده را بررسی می‌کنیم. مقاله [۴] شروع این روش‌ها بوده که این کار را با استفاده از نوعی مدل کرپسکی [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 ابتدایی‌ترین 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 چرا که این یک مسئله‌ی NP-Hard است. 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

| | | |
|----|---------------------------------------|----|
| ۱ | مقدمه و مفاهیم اولیه | ۱ |
| ۱ | ۱.۱ روش واریسی مدل | ۱ |
| ۳ | ۱.۱.۱ زبان LTL | ۳ |
| ۴ | ۲.۱.۱ معناشناسی LTL | ۴ |
| ۴ | ۲.۱ نحو زبان مورد بررسی | ۴ |
| ۶ | ۳.۱ معناشناسی زبان مورد بررسی | ۶ |
| ۷ | ۱.۳.۱ برچسب‌ها | ۷ |
| ۸ | ۲.۳.۱ رد پیشوندی | ۸ |
| ۹ | ۳.۳.۱ تعریف معناشناسی رد پیشوندی | ۹ |
| ۱۴ | ۲ صورت‌گیری جدید برای روش واریسی مدل | ۱۴ |
| ۱۴ | ۱.۲ ویژگی‌های برنامه‌ها | ۱۴ |
| ۱۵ | ۱.۱.۲ ویژگی‌ها | ۱۵ |
| ۱۵ | ۲.۱.۲ عبارات منظم | ۱۵ |
| ۱۶ | ۳.۱.۲ نحو عبارات منظم | ۱۶ |
| ۱۷ | ۴.۱.۲ معناشناسی عبارات منظم | ۱۷ |
| ۲۱ | ۵.۱.۲ گونه‌های مختلف زبان عبارات منظم | ۲۱ |
| ۲۲ | ۲.۲ صورت جدید مسئله‌ی واریسی مدل | ۲۲ |
| ۲۴ | ۳.۲ در مورد توقف پذیری | ۲۴ |

| | |
|----|----------------------------|
| ۳۰ | ۳ واری مدل منظم |
| ۳۰ | ۱.۳ در مورد عبارات منظم |
| ۳۰ | ۱.۱.۳ هم‌ارزی عبارات منظم |
| ۳۱ | ۲.۱.۳ فرم نرمال فصلی |
| ۳۷ | ۳.۱.۳ سر و دم عبارات منظم |
| ۴۳ | ۲.۳ واری مدل منظم |
| ۴۳ | ۱.۲.۳ صورت |
| ۴۶ | ۲.۲.۳ درستی و تمامیت |
| ۵۲ | ۴ واری مدل ساختارمند |
| ۵۹ | ۵ نتیجه‌گیری |
| ۶۰ | واژه‌نامه فارسی به انگلیسی |
| ۶۶ | واژه‌نامه انگلیسی به فارسی |

فصل ۱

مقدمه و مفاهیم اولیه

در این فصل به عنوان مقدمه، روش واریسی مدل به طور مختصر معرفی شده است. 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 واریسی مدل، بحث اصلی این پایان نامه شروع می‌شود. محوریت کار ما [۶] است که در آن روش جدید واریسی مدل ارائه شده است. بحث با ارائه‌ی نحوه‌ی یک زبان برنامه نویسی شروع می‌شود، سپس معناشناسی این زبان، یعنی معناشناسی رد پیشوندی^۱ 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 واریسی مدل

روش واریسی مدل یک روش صوری است که برای درستی‌یابی سیستم‌های مختلف استفاده می‌شود. در این روش معمولاً 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.

¹Prefix 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. مثلاً 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 واریسی مدل برای اطمینان از عدم وجود چنین ویژگی نامطلوبی استفاده کرد. مثال‌های 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 واریسی 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 غالباً در زبان طبیعی نقش قید را دارند. منطق‌های زمانی دسته‌ای از منطق‌های موجهات هستند که به صورتی‌گری ما مفهوم زمان را اضافه می‌کنند، یعنی قیدهایی مانند فعلاً، بعداً، و قبلاً (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 واریسی مدل استفاده می‌شود. البته در مورد قیدهایی 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، هر چند 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.

^۲Linear Temporal Logic

۱.۱.۱ زبان LTL

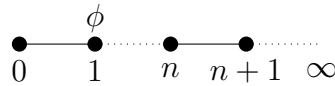
تعریف ۱.۱. هر عضو مجموعه Φ یک فرمول در زبان LTL است و Π مجموعه‌ی (شمارای نامتناهی) فرمول‌های اتمی است و $\pi \in \Pi$:

$$\Pi \subset \Phi,$$

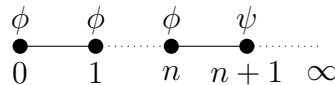
$$\phi \in \Phi \Leftrightarrow \phi ::= \pi \mid \phi \vee \phi \mid \neg \phi \mid \bigcirc \phi \mid \phi \mathcal{U} \phi$$

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. مسلماً 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 \neg, \vee هستند، به همان معنایی که در منطق گزاره‌ای کلاسیک داشتند. در ادوات جدید $\bigcirc \phi$ به معنای برقرار بودن این فرمول دقیقاً در لحظه‌ی بعدی (دقیقاً یک لحظه) است، مثلاً 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. (مثلاً 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 ادوات 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 معنانشناسی LTL

مدل‌های این منطق را به صورت توابع $M : \mathbb{N}_0 \rightarrow P(\Pi)$ 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. مثلاً، در مدلی به نام M در واقع $M(5)$ مجموعه‌ای 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 $M, i \models \phi$ نشان می‌دهیم و $M, i \models \phi$ به این معنی است که فرمول ϕ ، در لحظه‌ی i در مدل M درست است. این مفهوم را، به صورت بازگشتی، به شکل زیر تعریف می‌کنیم:

$$M, i \models \pi \text{ iff } \pi \in M(i)$$

$$M, i \models \neg\phi \text{ iff } M, i \not\models \phi$$

$$M, i \models \phi \vee \psi \text{ iff } M, i \models \phi \text{ or } M, i \models \psi$$

$$M, i \models \bigcirc\phi \text{ iff } M, i+1 \models \phi$$

$$M, i \models \phi \mathcal{U} \psi \text{ iff } \exists k \geq i \in \mathbb{N}_0 : \forall i \leq j < k : M, j \models \phi \text{ and } M, k \models \psi$$

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:

$$x, y, \dots \in \mathbb{X}$$

$$A \in \mathbb{A} ::= 1 \mid x \mid A_1 - A_2$$

$$B \in \mathbb{B} ::= A_1 < A_2 \mid B_1 \text{ nand } B_2$$

$$E \in \mathbb{E} ::= A \mid B$$

$$S \in \mathbb{S} ::=$$

$$x \doteq A;$$

$$| \ ;$$

$$| \text{ if } (B) \ S \mid \text{ if } (B) \ S \text{ else } S$$

$$| \text{ while } (B) \ S \mid \text{ break};$$

$$| \ \{SI\}$$

$$SI \in \mathbb{SL} ::= SI \ S \mid \epsilon$$

$$P \in \mathbb{P} ::= SI$$

قابل مشاهده است که این زبان، نسبت به کل زبان C ، تا حد ممکن ساده‌سازی شده است. علت این کار را بعداً عمیق‌تر حس خواهیم کرد. علت ساده‌تر شدن کار برای ارائه‌ی معنانشناسی و صورت‌های جدید روش واریسی مدل است. در اینجا، راحتی برنامه نوشتن در این زبان مطرح نبوده است، چون اصلاً این زبان برای این کار ساخته نشده است. هدف از ارائه‌ی این زبان صرفاً ارائه‌ی روش جدید است. یعنی می‌توان به این زبان به چشم یک مدل محاسبه، مانند ماشین تورینگ^۳ و ماشین رجیستر^۴ نگاه کرد. روشی که سعی در ارائه‌ی آن داریم، برای زبان‌های برنامه نویسی دستوری است، مانند پایتون^۵، جاوا^۶ 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. مثلاً ابتدا ۰ را به کمک ۱-۱ می‌سازیم و سپس، با استفاده از ۰ می‌توانیم یکی

³Turing Machine

⁴Register Machine

⁵Python

⁶Java

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 قابل ساختن هستند. در عبارت‌های بولی^۷ نیز با داشتن دسته‌ای محدود اما کافی از عملگرها، باقی عملگرهای رایج قابل بیان هستند. یعنی اینجا صرفاً ادات شفر^۸ تعریف شده است و باقی عملگرهای بولی^۹ 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 می‌توان ثابت کرد که این زبان هم‌ارز با ماشین تورینگ^[۹] است. هرآنچه بالاتر در مورد معنای دستورات این زبان گفتیم، به هیچ وجه صوری نبود. صرفاً 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 خواهیم کرد. بدیهی است که یک بیان صوری از روی یک درک شهودی ساخته می‌شود، اما این دو استفاده‌های متفاوتی دارند.

۳.۱ معنانشناسی زبان مورد بررسی

معنانشناسی زبانی را که در بخش پیش آورده‌ایم، با کمک مفاهیمی به نام‌های برجسب^{۱۰} و رد پیشوندی^{۱۱} و عملگری به نام چسباندن^{۱۲} تعریف می‌کنیم. نام این معنانشناسی، معنانشناسی رد پیشوندی است.

⁷Boolean Expression

⁸Sheffer Operator

⁹Boolean Operator

¹⁰Label

¹¹Prefix Trace

¹²Concatenation

۱.۳.۱ برچسب‌ها

با وجود 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] آورده شده‌اند. از آوردن مستقیم این تعاریف در اینجا خودداری کرده‌ایم. البته در مورد تعریف صوری برچسب‌ها، 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.

at[S] : برچسب شروع S.

aft[S] : برچسب پایان S، اگر پایانی داشته باشد.

esc[S] : یک مقدار بولی را باز می‌گرداند که بسته به اینکه در S عبارت دستوری break وجود دارد یا خیر، مقدار درست یا غلط را برمی‌گرداند.

brk-to[S] : برچسبی است که اگر حین اجرای S عبارت دستوری break اجرا شود، برنامه از آن برچسب ادامه پیدا می‌کند.

brks-of[S] : مجموعه‌ای از برچسب‌های عبارت‌های دستوری break که داخل S هستند را برمی‌گرداند.

in[S] : مجموعه‌ای از تمام برچسب‌های درون S را برمی‌گرداند.

labs[S] : مجموعه‌ای از تمام برچسب‌هایی که با اجرای S قابل دسترسی هستند را برمی‌گرداند.

¹³Statement

¹⁴Boolean 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 $\rho : \mathbb{X} \rightarrow \mathbb{V}$ را یک محیط می‌گوییم. مجموعه‌ی همه‌ی محیط‌ها را با $\mathbb{E}\mathbb{V}$ 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 \mathbb{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. 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 \mathbb{S}^+ و مجموعه‌ی ردهای پیشوندی نامتناهی را با \mathbb{S}^∞ نمایش می‌دهیم. مجموعه‌ی همه‌ی ردهای پیشوندی را هم با $\mathbb{S}^{+\infty}$ 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 \blacktriangleleft نشان داده‌ایم. در اثبات‌ها، برای هر حالت از \blacktriangleright استفاده کرده‌ایم.

¹⁵State

¹⁶Environment

¹⁷Value

¹⁸Variable

¹⁹Workflow

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 لزوماً 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 تعریف معناسازی رد پیشوندی

در این بخش، دو تابع A و B را به ترتیب روی عبارت‌های حسابی^{۲۰} و عبارت‌های بولی، یعنی A ها و B ها تعریف می‌کنیم، سپس با کمک آنها تابع S^* را روی مجموعه‌ای از اجتماع معنای S ها و SA ها تعریف می‌کنیم. به هر SA لیست عبارت‌های دستوری^{۲۱} می‌گوییم. پس در نهایت، هدف ما تعریف S^* است.

²⁰Arithmetic Expression

²¹Statement List

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

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

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

$$\blacktriangleleft \mathcal{A}[A_1 - A_2]\rho = \mathcal{A}[A_1]\rho - \mathcal{A}[A_2]\rho$$

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

$$\blacktriangleleft \mathcal{B}[A_1 < A_2]\rho = True \quad \text{اگر } \mathcal{A}[A_1]\rho \text{ کوچکتر از } \mathcal{A}[A_2]\rho \text{ باشد}$$

$$\blacktriangleleft \mathcal{B}[A_1 < A_2]\rho = False \quad \text{اگر } \mathcal{A}[A_1]\rho \text{ بزرگتر از } \mathcal{A}[A_2]\rho \text{ باشد}$$

$$\blacktriangleleft \mathcal{B}[B_1 \text{ nand } B_2]\rho = \neg(\mathcal{B}[B_1]\rho \wedge \mathcal{B}[B_2]\rho)$$

طبیعتاً \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 \mathbb{PUSIUS} ، لیست عبارت دستوری \mathbb{SI} و برنامه‌ی P انجام می‌دهیم. یعنی دامنه‌ی \mathcal{S}^* مجموعه‌ی \mathbb{PUSIUS} است. پیش از ادامه‌ی بحث، باید این نکته را درمورد علامت‌گذاری‌هایمان ذکر کنیم که منظور از $S ::= l \text{ break};$ این است که تاکید کرده‌ایم که S با برجسب l شروع شده‌است، هرچند که همین طور که پیش‌تر گفته‌شد، l جزو زبان نیست.

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

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

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

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

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

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

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

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

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

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

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

اگر $SI ::= \text{d}$ باشد، ردهای پیشوندی متناظر با اجرای این عبارت دستوری را به شکل مجموعه‌ی زیر تعریف می‌کنیم:

$$\mathcal{S}[SI] = \{\langle at[SI], \rho \rangle \mid \rho \in \mathbb{EV}\}$$

اگر $SI ::= SI' \ S$ باشد، ردهای پیشوندی متناظر با اجرای این عبارت دستوری را به شکل مجموعه‌ی زیر تعریف می‌کنیم:

$$\mathcal{S}[SI] = \mathcal{S}[SI'] \cup (\mathcal{S}[SI'] \bowtie \mathcal{S}[S])$$

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:

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

اگر $S ::= \text{while}(B)S_b$ باشد، ماجرنا نسبت به حالات قبل اندکی پیچیده‌تر می‌شود. تابعی به اسم \mathcal{F} را تعریف خواهیم کرد 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 می‌گیرد 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 می‌کند و دنباله‌هایی جدید را از دنباله‌های قبلی می‌سازد. معنای یک حلقه را کوچکترین نقطه ثابت^{۲۲} این تابع در نظر می‌گیریم. در ادامه، تعریف \mathcal{F} آمده است. با دیدن تعریف، می‌توان به دلیل این کار پی برد. 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، نقطه ثابت^{۲۳} مفهوم مناسبی برای این است که از آن در تعریف صوری معنای حلقه استفاده شود. علت اینکه کوچکترین نقطه ثابت را به عنوان معنای حلقه در نظر می‌گیریم، این است که مطمئن هستیم، هر رد پیشوندی‌ای در نقطه ثابت 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 کاملاً بی‌ربط به گردش کار برنامه به ته ردهای پیشوندی، که صرفاً برچسب متفاوتی با آخرین وضعیت هر رد پیشوندی دارند، نقطه ثابت جدیدی ساخته‌ایم. پس اگر خودمان را محدود به انتخاب کوچکترین نقطه ثابت نکنیم، به توصیفات صوری خوبی از برنامه‌ها دست پیدا نخواهیم کرد.

در مورد نقطه ثابت این نکته باقی می‌ماند که چه طور می‌توانیم مطمئن باشیم که چنین نقطه ثابتی 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 \mathcal{S}^* یکنوا است، بنا به قضیه‌ی نسترو-تارسکی^{۲۴} [۲۳] برای تابع \mathcal{S}^* کوچکترین نقطه ثابت وجود دارد. تعاریف توابعی که درموردشان

²²Least Fixpoint

²³Fixpoint

²⁴Knaster-Tarski Theorem

صحبت کردیم، به این شکل است:

$$\begin{aligned}\mathcal{S}[\![S]\!] &= lfp^{\subseteq} \mathcal{F}[\![S]\!] \\ \mathcal{F}[\![S]\!]X &= \{\langle at[\![S]\!], \rho \rangle \mid \rho \in \mathbb{EV}\} \cup \\ &\quad \{\pi_2 \langle l, \rho \rangle \langle aft[\![S]\!], \rho \rangle \mid \pi_2 \langle l, \rho \rangle \in X \wedge \mathcal{B}[\![B]\!] \rho = False \wedge l = at[\![S]\!]\} \cup \\ &\quad \{\pi_2 \langle l, \rho \rangle \langle at[\![S_b]\!], \rho \rangle \pi_3 \mid \pi_2 \langle l, \rho \rangle \in X \wedge \mathcal{B}[\![B]\!] \rho = True \wedge \\ &\quad \langle at[\![S_b]\!], \rho \rangle \pi_3 \in \mathcal{S}[\![S_b]\!] \wedge l = at[\![S]\!]\}\end{aligned}$$

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

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

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

$$\mathcal{S}[\![S]\!] = \mathcal{S}[\![SI]\!]$$

در اینجا، تعریف معنای برنامه‌ها به پایان می‌رسد.

فصل ۲

صوری‌گری جدید برای روش واریسی مدل

در این فصل، صورت جدیدی برای روش واریسی 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 واریسی مدل را ارائه می‌کنیم. در آخر این فصل نیز، به بحث در مورد تصمیم‌پذیری این روش می‌پردازیم.

۱.۲ ویژگی‌های برنامه‌ها

تا به اینجای کار، نحو و معناشناسی یک زبان برنامه‌نویسی را تعریف کرده‌ایم. در این بخش، در مورد ویژگی‌های برنامه‌هایی که در این زبان نوشته می‌شوند، با توجه به معناشناسی رد‌پیشوندی که در فصل پیش تعریف کردیم، صحبت می‌کنیم. برای برنامه‌هایی که در یک زبان برنامه‌نویسی نوشته می‌شوند، می‌توان به اشکال مختلفی ویژگی تعریف کرد. مثلاً ویژگی‌های کاملاً 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 عموماً 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 $S^*[P]$ 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. می‌توانیم 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 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 $P \rightarrow P(S^*)$ است. پس یعنی یک برنامه را در ورودی می‌گیرد و یک مجموعه از ردهای پیشوندی را باز می‌گرداند. پس می‌توانیم، هر ویژگی را به عنوان زیرمجموعه‌ای از $P(S^*)$ تعریف کنیم، به عبارت دیگر عضوی از $P(P(S^*))$. 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 واریسی مدل از منطق‌های 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 واریسی 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 واریسی مدل برای درستی‌یابی برنامه‌های کامپوتری 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^۱هاست. 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 $L : B$ نمایش می‌دهیم.

نحو عبارات منظم به شکل زیر است:

تعریف ۱.۲.

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

$$x, y, \dots \in \mathbb{X}$$

$$\underline{x}, \underline{y}, \dots \in \underline{\mathbb{X}}$$

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

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

$$R ::= \varepsilon$$

$$| L : B$$

$$| R_1 R_2 \text{ (or } R_1 \bullet R_2)$$

$$| R_1 \mid R_2 \text{ (or } R_1 + R_2)$$

$$| R_1^*$$

$$| R_1^+$$

$$| (R_1)$$

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

^۱Atom

^۲Regular Expressions Semantics

خواهیم دید که عملگر یگانی $+$ به وسیله‌ی عملگر یگانی $*$ قابل بیان است. توجه شود که پرانترها 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 $B : ?$ همان عبارت منظم $B : \mathbb{L}$ است. عبارت منظم $B : l$ به جای عبارت منظم $B : \{l\}$ به کار می‌رود و منظور از عبارت منظم $B : l$ نیز عبارت منظم $B : \mathbb{L} \setminus \{l\}$ است.

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

تعریف ۲.۲. (محیط اولیه^۳): به هر تابع $\rho : \mathbb{X} \rightarrow \mathbb{V}$ یک محیط اولیه می‌گوییم. مجموعه‌ی همه‌ی محیط‌های اولیه را با \mathbb{EV} نمایش می‌دهیم.

۴.۱.۲ معناسازی عبارات منظم

معنای عبارات منظم را با استفاده از تابع S^r نشان می‌دهیم. این تابع به این شکل تعریف می‌شود که در ورودی یک عبارت منظم R را می‌گیرد، سپس یک مجموعه از زوج‌های $\langle \rho, \pi \rangle$ را که $\pi \in \mathbb{G}^*$ و $\rho \in \mathbb{EV}$ باز می‌گرداند. بنابراین این تابع از نوع $\mathbb{R} \rightarrow P(\mathbb{EV} \times \mathbb{G}^*)$ است. منظور از \mathbb{G}^* نیز $\mathbb{G}^+ \cup \{\epsilon\}$ است.

تعریف استقرایی تابع S^r به شکل زیر است:

تعریف ۳.۲. تابع $S^r : \mathbb{R} \rightarrow P(\mathbb{EV} \times \mathbb{G}^*)$ به صورت استقرایی روی ساختار عبارت منظم R به صورت زیر تعریف می‌شود:

$$S^r[\epsilon] = \{\langle \rho, \epsilon \rangle \mid \rho \in \mathbb{EV}\}$$

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

$$S^r[\mathbb{L} : B] = \{\langle \rho, \langle l, \rho \rangle \rangle \mid l \in \mathbb{L} \wedge B[B]\rho, \rho\}$$

³Initial Environment

[این یعنی معنای عبارت منظم $L : B$ مجموعه‌ای است شامل زوج مرتب‌هایی که عضو اول آن‌ها محیط‌های اولیه 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 ρ و محیط ρ برقرار باشد. با توجه به حضور محیط‌های اولیه، در اینجا B به جای اینکه از نوع $\mathbb{E}V \rightarrow \mathbb{B}OOL$ باشد، از نوع $\mathbb{E}V \rightarrow \mathbb{E}V \rightarrow \mathbb{B}OOL$ است (منظور از $\mathbb{B}OOL$ همان مجموعه‌ی $\{T, F\}$ است). بعد از این تعریف، A و B را با در نظر گرفتن محیط اولیه دوباره تعریف خواهیم کرد.]

$$S^r[[R_1 R_2]] = S^r[[R_1]] \bowtie S^r[[R_2]]$$

به طوریکه، با فرض اینکه دو مجموعه‌ی S و S' هر یک معنای یک عبارت منظم باشند:

$$S \bowtie S' = \{ \langle \rho, \pi \pi' \rangle \mid \langle \rho, \pi \rangle \in S \wedge \langle \rho, \pi' \rangle \in S' \}$$

[این یعنی اگر یک عبارت منظم داشته باشیم که از چسباندن R_1 و R_2 به هم ساخته شده باشد، آنگاه معنای این عبارت منظم از زوج‌هایی تشکیل شده است که مولفه‌ی اول آن‌ها محیط‌های اولیه هستند و مولفه‌ی دوم آن‌ها از چسباندن ردهای پیشوندی موجود در مولفه‌ی دوم اعضای مجموعه‌ی معنای این دو عبارت منظم تشکیل شده است. عملگر $Join$ که برای معنای عبارات منظم تعریف شده است، با تعریف عملگر چسباندن معنای دو برنامه متفاوت است. مورد دوم را در فصل قبل داشتیم که با کمک \bowtie روی ردهای پیشوندی تعریف می‌شد، اما در تعریفی که در اینجا از \bowtie ارائه شده است، از عملگر \bowtie روی ردهای پیشوندی استفاده نشده است.]

تا این قسمت از تعریف معنای عبارت منظم که رسیده‌ایم، تا حدی به درکی شهودی از اینکه به چه نحوی قرار است عبارات منظم راهی برای توصیف ویژگی در مورد برنامه‌ها باشند، نزدیک‌تر شده‌ایم. همان‌طور که در مورد قبل دیدیم، هر زوج $L : B$ دقیقاً به یک وضعیت داخل یک رد پیشوندی اشاره می‌کند. انگار که قرار است این زوج‌ها موازی با وضعیت‌ها در ردهای پیشوندی موجود در معنای یک برنامه جلو روند و انطباق را بررسی کنند تا واریسی مدل انجام شود. درک این موضوع اولین قدم ما در دیدن عصاره‌ی روش واریسی مدل است، در ادبیاتی که از شروع فصل دوم به پا کرده‌ایم.]

$$S^r[[R_1 \mid R_2]] = S^r[[R_1]] \cup S^r[[R_2]]$$

[این مورد، 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.]

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

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

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

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

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

[این دو عبارت هم تعریف معنای دو عملگر * و + هستند. منظور از \mathbb{N} مجموعه‌ی اعداد طبیعی است. همان‌طور که قبل‌تر هم اشاره شد + را می‌توان با * تعریف کرد. اضافه می‌کنیم که * را در فرازبان (و نه در زبان عبارات منظم) می‌توان با عملگر انتخاب تعریف کرد.]

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

[این قسمت از تعریف هم صرفاً بیان می‌کند که پرانتزها تاثیری در معنای عبارات منظم ندارند که کاملاً قابل انتظار است، چرا که وجود پرانتز قرار است صرفاً در خواص نحوی زبان اثر بگذارد.]

تعریف معنای عبارات منظم در اینجا تمام می‌شود، اما همان‌گونه که در لابه‌لای تعاریف گفته‌شد، احتیاج داریم که A و B را از نو تعریف کنیم:

تعریف ۴.۲. توابع $A : \mathbb{A} \rightarrow \underline{\mathbb{E}\mathbb{V}} \rightarrow \mathbb{E}\mathbb{V} \rightarrow \mathbb{B}\mathbb{O}\mathbb{O}\mathbb{L}$ و $B : \mathbb{B} \rightarrow \underline{\mathbb{E}\mathbb{V}} \rightarrow \mathbb{E}\mathbb{V} \rightarrow \mathbb{V}$ به شکل استقرایی به ترتیب روی ساختارهای $A \in \mathbb{A}$ و $B \in \mathbb{B}$ به شکل زیر تعریف می‌شوند:

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

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

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

$$\mathcal{A}[A_1 - A_2]_{\underline{\rho}}, \rho = \mathcal{A}[A_1]_{\underline{\rho}}, \rho - \mathcal{A}[A_2]_{\underline{\rho}}, \rho$$

$$\mathcal{B}[A_1 < A_2]_{\underline{\rho}}, \rho = \mathcal{A}[A_1]_{\underline{\rho}}, \rho < \mathcal{A}[A_2]_{\underline{\rho}}, \rho$$

$$\mathcal{B}[B_1 \text{ and } B_2]_{\underline{\rho}}, \rho = \mathcal{B}[B_1]_{\underline{\rho}}, \rho \uparrow \mathcal{B}[B_2]_{\underline{\rho}}, \rho$$

به راحتی قابل مشاهده است که تعاریف جدید تا حد خوبی به تعاریف قبلی شبیه هستند و فرق عمده صرفاً وارد شدن ρ است.

حال که معنانشناسی 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 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 نامعادل. هدف از ارائه‌ی جبر کلاینی این بوده است که تعمیمی باشد که این تعاریف نابرابر را در خود جای می‌دهد. در [۱۶] آمده است که برای جبر کلاینی هم تعاریف متفاوتی که با هم برابر 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 $\mathcal{S}^r[\varepsilon R] = \mathcal{S}^r[\varepsilon]$ ، چون ε نقش صفر عملگر چسبانند (که عملگر ضرب جبر کلاینی است) را دارد و یک جبر کلاینی برای صفر خاصیت جذب را به عنوان یک اصل دارد. اما در مورد عبارات منظمی که در اینجا تعریف کردیم داریم $\mathcal{S}^r[\varepsilon R] = \mathcal{S}^r[R]$. 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 چسبانند است و بعد به ادامه‌ی راه اصلیمان می‌پردازیم.

قضیه ۵.۲. برای عبارات منظم R, R_1, R_2 داریم:

$$\mathcal{S}^r[R \bullet (R_1 + R_2)] = \mathcal{S}^r[(R \bullet R_1) + (R \bullet R_2)]$$

اثبات.

$$\begin{aligned} \mathcal{S}^r[R \bullet (R_1 + R_2)] &= \mathcal{S}^r[R] \bowtie \mathcal{S}^r[(R_1 + R_2)] \\ &= \mathcal{S}^r[R] \bowtie \mathcal{S}^r[R_1 + R_2] = \mathcal{S}^r[R] \bowtie (\mathcal{S}^r[R_1] \cup \mathcal{S}^r[R_2]) \\ &= \{ \langle \underline{\rho}, \pi \pi' \rangle \mid \langle \underline{\rho}, \pi \rangle \in \mathcal{S}^r[R] \wedge (\langle \underline{\rho}, \pi' \rangle \in \mathcal{S}^r[R_1] \vee \langle \underline{\rho}, \pi' \rangle \in \mathcal{S}^r[R_2]) \} \\ &= \{ \langle \underline{\rho}, \pi \pi' \rangle \mid (\langle \underline{\rho}, \pi \rangle \in \mathcal{S}^r[R] \wedge \langle \underline{\rho}, \pi' \rangle \in \mathcal{S}^r[R_1]) \vee (\langle \underline{\rho}, \pi \rangle \in \mathcal{S}^r[R] \wedge \langle \underline{\rho}, \pi' \rangle \in \mathcal{S}^r[R_2]) \} \\ &= \{ \langle \underline{\rho}, \pi \pi' \rangle \mid \langle \underline{\rho}, \pi \rangle \in \mathcal{S}^r[R] \wedge \langle \underline{\rho}, \pi' \rangle \in \mathcal{S}^r[R_1] \} \cup \{ \langle \underline{\rho}, \pi \pi' \rangle \mid \langle \underline{\rho}, \pi \rangle \in \mathcal{S}^r[R] \wedge \langle \underline{\rho}, \pi' \rangle \in \mathcal{S}^r[R_2] \} \end{aligned}$$

$$\begin{aligned}
&= (\mathcal{S}^r[R] \bowtie \mathcal{S}^r[R_1]) \cup (\mathcal{S}^r[R] \bowtie \mathcal{S}^r[R_2]) = \mathcal{S}^r[R \bullet R_1] \cup \mathcal{S}^r[R \bullet R_2] \\
&= \mathcal{S}^r[(R \bullet R_1) + (R \bullet R_2)]
\end{aligned}$$

□

تا اینجا کار، بیشتر مفاهیمی که برای بیان صورت جدید مسئله‌ی واری مدلی احتیاج داریم را بیان کرده‌ایم.

۵.۱.۲ گونه‌های مختلف زبان عبارات منظم

به عنوان قسمت آخر این بخش، گونه‌های مختلفی از زبان عبارات منظم را بیان می‌کنیم که هر کدام در واقع زیرمجموعه‌ای از کل زبانی 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. اولین گونه‌ای که می‌خواهیم بیان کنیم، گونه‌ای است که در اعضای آن اصلاً عبارت $L : B$ حضور ندارد و کل عبارت‌های زبان از ε ها تشکیل شده‌اند.

تعریف ۶.۲. (عبارت منظم تهی - \mathbb{R}_ε):

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

$$R ::= \varepsilon \mid R_1 R_2 \mid R_1 + R_2 \mid R_1^* \mid R_1^+ \mid (R_1)$$

با توجه به بخش قبل، معنای همه‌ی این عبارت‌ها برابر $\{\langle \rho, \varepsilon \rangle\}$ خواهد بود. گونه‌ی بعدی عبارت منظم ناتهی است.

تعریف ۷.۲. (عبارت منظم ناتهی - \mathbb{R}^+):

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

$$R ::= L : B \mid \varepsilon R_2 \mid R_1 \varepsilon \mid R_1 R_2 \mid R_1 + R_2 \mid R_1^+ \mid (R_1)$$

دلیل وجود εR_2 و $R_1 \varepsilon$ در تعریف این است که ممکن است معنای عبارتی در این زبان با $\langle \rho, \varepsilon \rangle$ برابر نباشد، اما در خود عبارت، ε حضور داشته باشد. با این تفصیل می‌توان دید که دو مجموعه‌ی \mathbb{R}_ε و \mathbb{R}^+ یک افزاز برای مجموعه‌ی \mathbb{R} هستند، چونکه معنای هر عبارت در \mathbb{R} یا با $\langle \rho, \varepsilon \rangle$ برابر

هست یا نیست. بنابراین شاید به نظر برسد که تعریف یکی از آن‌ها به طور ساختاری کافی باشد. اما این طور نیست، چون ممکن است که درجایی احتیاج داشته باشیم که تعریفی استقرایی روی هر یک از این دو زبان 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 است.

تعریف ۸.۲. (عبارت منظم بدون انتخاب - \mathbb{R}^\dagger):

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

$$R ::= \varepsilon \mid L : B \mid R_1 R_2 \mid R_1^* \mid R_1^+ \mid (R_1)$$

۲.۲ صورت جدید مسئله‌ی واریسی مدل

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

پیش از ارائه‌ی تعریف واریسی مدل، نیاز داریم که عملگر بستر پیشوندی را برای یک مجموعه از ردهای پیشوندی معرفی کنیم.

تعریف ۹.۲. (بستر پیشوندی): اگر $\Pi \in P(\mathbb{E}\mathbb{V} \times \mathbb{G}^+)$ ، آنگاه بستر پیشوندی Π را به صورت زیر تعریف می‌کنیم:

$$\text{prefix}(\Pi) = \{ \langle \underline{\rho}, \pi \rangle \mid \pi \in \mathbb{G}^+ \wedge \exists \pi' \in \mathbb{G}^* : \langle \underline{\rho}, \pi \pi' \rangle \in \Pi \}$$

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

مثال ۱۰.۲. 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:

$$\begin{aligned} \text{prefix}(\Pi) = & \{ \langle \underline{\rho}, \langle l_1, \rho_1 \rangle \rangle, \langle \underline{\rho}, \langle l_1, \rho_1 \rangle \langle l_2 \rho_2 \rangle \rangle, \langle \underline{\rho}, \langle l_1, \rho_1 \rangle \langle l_2 \rho_2 \rangle \langle l_3, \rho_3 \rangle \rangle, \\ & \langle \underline{\rho}, \langle l_1', \rho_1' \rangle \rangle, \langle \underline{\rho}, \langle l_1', \rho_1' \rangle \langle l_2' \rho_2' \rangle \rangle \} \end{aligned}$$

که شامل ۵ عضو است.

حال به ارائه‌ی صورت جدیدمان از روش واریسی مدل می‌رسیم.

تعریف ۱۱.۲. (واریسی مدل): اگر $\rho \in \mathbb{EV}$, $R \in \mathbb{R}^+$, $P \in \mathbb{P}$ آنگاه:

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

این تعریف بیان می‌کند که اگر برنامه‌ی P با محیط اولیه‌ی ρ اجرا شود، این برنامه در صورتی ویژگی‌ای که با یک عبارت منظم R بیان شده است را دارد که معنای آن زیرمجموعه‌ی بستار پیشوندی آن معنای عبارت منظم $R \bullet (? : T)^*$ باشد. توجه شود که محیط اولیه‌ای که برای برنامه‌ی مورد بررسی متصور هستیم، صرفاً به این منظور در تعریف قرار داده شده است که بتوانیم، معنای برنامه‌ها را با معنای عبارات منظم قابل قیاس کنیم. دلیل حضور محیط اولیه در معنای عبارات منظم نیز در صورت سوم روش واریسی مدل مشخص می‌شود، یعنی جایی که واریسی مدل روی ساختار برنامه تعریف شده است و ردهای پیشوندی موجود در هر قسمت از برنامه با محیط متفاوتی شروع می‌شوند و اطلاعات محیط اولیه‌ی برنامه در این ردها حضور ندارد (با اینکه ممکن است به آن نیاز داشته باشیم). در این صورت از روش واریسی مدل و صورت بعدی، محیط‌های اولیه صرفاً حضور دارند و در عمل نقش مهمی ندارند.

در مورد نقش $(? : T)^*$ و prefix نیز می‌توان گفت، بنا به تصمیم مبدع این روش، اگر یک رد پیشوندی، همه‌ی زوج‌های $L : B$ را ارضا کند و بدون اینکه با هیچ کدام از آن‌ها ناسازگاری‌ای داشته باشد به اتمام برسد، درحالی‌که هنوز عبارت منظم به اتمام نرسیده است، این رد پیشوندی با ویژگی بیان شده با عبارت منظم را دارد. این نکته صرفاً در مورد حضور prefix بود. حضور $(? : T)^*$ نیز باعث می‌شود اگر طول عبارت منظم مورد بررسی کمتر از رد پیشوندی بود و ناسازگاری‌ای مشاهده نشده بود، رد پیشوندی دارای ویژگی R در نظر گرفته شود.

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 کل کاری که در حال توصیفش هستیم، قابل پیاده سازی نیست!

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

تعریف ۱۲.۲. (توقف پذیری): برنامه‌ی P را به همراه اجرای اولیه ρ توقف پذیر می‌گوییم، اگر و تنها اگر وجود داشته باشد $\pi \in \mathcal{S}^*[P]$ ، که ρ محیط متناظر با محیط اولیه‌ی ρ است):

$$\pi = \langle at[P], \rho \rangle \pi'$$

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

در این تعریف، توقف پذیری صرفاً برای یک محیط اولیه تعریف شده است. در اینجا، توقف پذیری به متناهی بودن ردهای پیشوندی موجود در برنامه ارتباطی ندارد. با توجه به معناشناسی‌ای که داریم، 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 حتماً 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 آورده‌ایم، مطمئن‌تر هستیم. با این حال، در قضیه‌ی بعدی می‌بینیم که تعریفی که ارائه کردیم با اینکه بگوییم، در برنامه رد پیشوندی نامتناهی وجود ندارد معادل است.

قضیه ۱۳.۲. برای برنامه‌ی P و محیط اولیه‌ی ρ داریم $P, \rho \downarrow$ ، اگر و تنها اگر ρ محیط متناظر با محیط اولیه‌ی ρ باشد و

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

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

حکم را با استقرا روی ساختار SI ثابت می‌کنیم.

► $SI = \varepsilon$:

داریم:

$$\mathcal{S}^*[\mathfrak{e}] = \{\langle at[\mathfrak{e}], \dot{\rho} \rangle \mid \dot{\rho} \in \mathbb{EV}\}$$

همین طور طبق تعریف برچسب‌ها داریم:

$$at[\mathfrak{e}] = aft[\mathfrak{e}]$$

پس حکم برقرار است.

$$\blacktriangleright SI = SI' S :$$

اینکه در معنای SI ، دنباله‌ای شامل $\langle aft[SI], \rho' \rangle$ وجود داشته باشد، با توجه به تعاریفی که داریم، به این وابسته است که در معنای S ، دنباله‌ای شامل $\langle aft[S], \rho' \rangle$ 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.

$$\blacktriangleright\blacktriangleright S = x \doteq A; :$$

در این حالت، با توجه به تعریف معنای S که قبل‌تر ارائه شد، دنباله‌ی

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

در معنای عبارت دستوری به ازای هر ρ وجود دارد. پس دنباله‌ای به شکل بالا با محیطی متناظر با ρ هم در معنای عبارت دستوری حضور دارد.

$$\blacktriangleright\blacktriangleright S = ; :$$

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

$$\langle at[S], \rho \rangle \langle aft[S], \rho \rangle$$

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

در صورتی که $B[B]\rho = T$ برقرار باشد، دنباله‌ی

$$\langle at[S], \rho \rangle \langle at[S_t], \rho \rangle \pi$$

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

$$\langle at[S], \rho \rangle \langle aft[S], \rho \rangle$$

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

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:

$$\langle at[S], \rho \rangle \langle at[S_f], \rho \rangle \pi$$

همین طور، تساوی $aft[S_t] = aft[S] = aft[S_f]$ هم طبق تعریف برچسب‌ها برقرار است.

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

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 حلقه با استفاده از یک تابع تعریف می‌شود. معنای حلقه کوچکترین نقطه ثابت این تابع است، در حالیکه، این تابع وقتی روی یک مجموعه از ردهای پیشوندی اعمال شود، تاثیرات یک بار اجرای عبارت‌های دستوری درون حلقه را روی ردهای پیشوندی درون مجموعه اعمال می‌کند.

طبق تعریف \mathcal{F} ، مطمئن هستیم که رد پیشوندی‌ای که با محیط ρ شروع شود، در مجموعه‌ی معنای S وجود دارد، چون به ازای هر محیط ρ (نقطه به این خاطر است که با ρ خاص موجود در فرض اشتباه گرفته نشود) حالت $\langle at[S], \rho \rangle$ در هر اعمال تابع \mathcal{F} روی هر مجموعه‌ی دلخواه وجود دارد. چون معنای S را به عنوان کوچکترین نقطه ثابت \mathcal{F} در نظر گرفته‌ایم، مطمئن هستیم که مجموعه‌ای که کوچکترین نقطه ثابت تابع \mathcal{F} است، شامل رد پیشوندی $\langle at[S], \rho \rangle$ 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 $\langle aft[S], \rho' \rangle \pi \langle at[S], \rho \rangle$ 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 طول 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 $\langle at[S_t], \rho'' \rangle$ 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 تفصیل، این مورد هم ثابت می‌شود.

►► $S = \text{break}; :$

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 که در مورد بخش‌هایی از تعریف این توابع، که تعریف مشخصی ارائه نشده است، برداشت آزاد است! ما در اینجا سعی داریم، معقول‌ترین 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 S_t در حلقه‌ی زیر باشد،

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

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

از پس این فرض‌های ما $aft[\text{break};] = \text{break} - to[S_t]$ نتیجه می‌شود و طبق معنای عبارت دستوری $\text{break};$ ، 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.

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

در این صورت، توقف پذیری S' را از فرض استقرای، استقرای قبلی (روی لیست عبارت‌های دستوری) داریم، پس $\{S'\}$ 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 همین یک مورد بحث می‌کنیم.

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

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 $\langle at[S], \rho \rangle \pi \langle aft[S], \rho' \rangle \pi'$ وجود دارد. باید ثابت کنیم، به ازای π' دلخواه، اگر رد پیشوندی $\langle at[S], \rho \rangle \pi \langle aft[S], \rho' \rangle \pi'$ داخل $S^*[S]$ وجود داشته باشد، آنگاه $\pi' = \epsilon$ برقرار است.

اگر برچسب $aft[S]$ در یک وضعیت از رد پیشوندی‌ای که گفتیم، حضور داشته باشد، یعنی در یک دور از اجرای حلقه، عبارت بولی لزوماً ارزش غلط داشته است. این باعث شده است که یک وضعیت شامل برچسب aft به یک رد پیشوندی چسبانده بشود و در نتیجه، این رد پیشوندی ساخته شده است. از طرفی دیگر هم می‌دانیم که وقتی عبارت بولی حاضر در ساختار حلقه غلط شده باشد، دیگر به ردهای پیشوندی داخل معنای حلقه چیزی اضافه نمی‌شود. بنابراین حالت ممکن $\pi' = \epsilon$ 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 واریسی مدل عبارت منظم R را با عبارت منظم ε جایگزین کنیم داریم:

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

طبق تعریف معنای عبارات منظم، هر رد پیشوندی متناهی ای داخل مجموعه‌ی سمت راستی رابطه‌ی زیرمجموعه بودن قرار می‌گیرد. این یعنی اگر الگوریتمی برای بررسی $P, \rho \models R$ داشته باشیم، این الگوریتم می‌تواند تشخیص دهد که آیا برنامه‌ی P با محیط اولیه‌ی ρ 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 شیوه‌ای 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 بیانی ساختارمندتر برای روش واری مدل می‌رسیم. ساختاری که در این فصل به صورت روش واری مدل اضافه می‌شود، ساختار عبارات منظم است. از این رو، پیش از اینکه به بیان واری مدل منظم بپردازیم، نیاز داریم که ابتدا، به بررسی و تعریف برخی خواص عبارات منظم بپردازیم که در ادامه، برای بیان واری مدل مورد نیاز هستند.

۱.۳ در مورد عبارات منظم

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

۱.۱.۳ هم‌ارزی عبارات منظم

هم‌ارزی 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 اگر شرط زیر برقرار باشد:

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

این هم‌ارزی را با $R_1 \approx R_2$ نمایش می‌دهیم.

قضیه ۲.۳. هم‌ارزی \Leftrightarrow تعریف شده روی مجموعه‌ی عبارات منظم یک رابطه‌ی هم‌ارزی است.

اثبات. برای هر عبارت منظم R داریم:

$$\mathcal{S}^r[R] = \mathcal{S}^r[R] \Rightarrow R \Leftrightarrow R$$

پس این رابطه انعکاسی است.

اگر $R_1, R_2 \in \mathbb{R}$ آنگاه داریم:

$$\mathcal{S}^r[R_1] = \mathcal{S}^r[R_2] \rightarrow \mathcal{S}^r[R_2] = \mathcal{S}^r[R_1] \Rightarrow R_1 \Leftrightarrow R_2 \rightarrow R_2 \Leftrightarrow R_1$$

پس این رابطه تقارنی هم هست.

اگر $R_1, R_2, R_3 \in \mathbb{R}$ داریم:

$$\mathcal{S}^r[R_1] = \mathcal{S}^r[R_2] \wedge \mathcal{S}^r[R_2] = \mathcal{S}^r[R_3] \rightarrow \mathcal{S}^r[R_1] = \mathcal{S}^r[R_3]$$

$$\Rightarrow R_1 \Leftrightarrow R_2 \wedge R_2 \Leftrightarrow R_3 \rightarrow R_1 \Leftrightarrow R_3$$

□

پس این رابطه متعدی نیز هست.

۲.۱.۳ فرم نرمال فصلی

یک دسته از عبارات منظم هستند که به آن‌ها فرم نرمال فصلی می‌گوییم. در صورتی از واریسی 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 $R \in \mathbb{R}$ را یک فرم نرمال فصلی می‌گوییم، اگر و تنها اگر با فرض اینکه عبارات منظم بدون انتخاب $R_1, R_2, \dots, R_n \in \mathbb{R}^+$ وجود داشته باشند که $R = R_1 + R_2 + \dots + R_n$.

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:

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

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

$$\blacktriangleleft \text{dnf}(R_1 R_2) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} R_1^i R_2^j$$

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

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

$$\blacktriangleleft \text{dnf}(R^*) = ((R_1)^*(R_2)^*\dots(R_n)^*)^*$$

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

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

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

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

اثبات. از استقرا روی ساختار R استفاده می‌کنیم.

$$\blacktriangleright R = \varepsilon :$$

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

که ε یک فرم نرمال فصلی است.

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

$$\text{dnf}(L : B) = L : B$$

که $L : B$ هم یک فرم نرمال فصلی است.

► $R = R_1 R_2$:

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

$$\text{dnf}(R_1 R_2) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} R_1^i R_2^j$$

که طرف راست عبارت بالا یک ترکیب نرمال فصلی است، چون هر $R_1^i R_2^j$ یک عضو از \mathbb{R}^I است.

► $R = R_1 + R_2$:

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

► $R = R_1^*$:

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

$$\text{dnf}(R_1^*) = ((R_1^1)^* (R_1^2)^* \dots (R_1^n)^*)$$

که:

$$\text{dnf}(R_1) = R_1^1 + R_1^2 + \dots + R_1^n$$

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

► $R = R_1^+$:

طبق تعریف، داریم:

$$\text{dnf}(R_1^+) = \text{dnf}(R_1 R_1^*)$$

$$\text{dnf}(R_1^*) = ((R_1^1)^* (R_1^2)^* \dots (R_1^n)^*)$$

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

$$R_1 = R_1^1 + \dots + R_1^n$$

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

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

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

طبق تعریف، داریم:

$$\text{dnf}((R_1)) = (\text{dnf}(R_1))$$

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

□

گزاره‌ی دیگری که برای اثبات مانده است، برقرار بودن $R \approx \text{dnf}(R)$ 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:

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

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 $R \in \mathbb{R}$ داریم:

$$\text{dnf}(R) \approx R$$

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

► $R = \varepsilon$:

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

► $R = L : B$:

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

► $R = R_1 R_2$:

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

$$\mathcal{S}^r[R_1 R_2] \subseteq \mathcal{S}^r[\text{dnf}(R_1 R_2)]$$

$$\mathcal{S}^r[R_1 R_2] \supseteq \mathcal{S}^r[\text{dnf}(R_1 R_2)]$$

: (\supseteq)

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

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

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

در این صورت، داریم:

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

$$\Rightarrow \langle \underline{\rho}, \pi_1 \pi_2 \rangle = \langle \underline{\rho}, \pi \rangle \in \mathcal{S}^r[R_1 R_2]$$

: (\subseteq)

$$\langle \underline{\rho}, \pi \rangle \in \mathcal{S}^r[R_1 R_2]$$

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

طبق فرض استقرا، داریم:

$$\mathcal{S}^r \llbracket R_1 \rrbracket = \mathcal{S}^r \llbracket \text{dnf}(R_1) \rrbracket, \mathcal{S}^r \llbracket R_2 \rrbracket = \mathcal{S}^r \llbracket \text{dnf}(R_2) \rrbracket,$$

$$\Rightarrow \exists k_1, k_2 : \langle \underline{\rho}, \pi_1 \rangle \in \mathcal{S}^r \llbracket R_1^{k_1} \rrbracket, \langle \underline{\rho}, \pi_2 \rangle \in \mathcal{S}^r \llbracket R_2^{k_2} \rrbracket$$

$$\Rightarrow \langle \underline{\rho}, \pi \rangle \in \mathcal{S}^r \llbracket R_1^{k_1} R_2^{k_2} \rrbracket \subseteq \mathcal{S}^r \llbracket \text{dnf}(R_1 R_2) \rrbracket$$

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

$$\mathcal{S}^r \llbracket \text{dnf}(R_1 + R_2) \rrbracket =$$

$$\mathcal{S}^r \llbracket \text{dnf}(R_1) + \text{dnf}(R_2) \rrbracket =$$

$$\mathcal{S}^r \llbracket \text{dnf}(R_1) \rrbracket \cup \mathcal{S}^r \llbracket \text{dnf}(R_2) \rrbracket =$$

(به کمک فرض استقرا)

$$\mathcal{S}^r \llbracket R_1 \rrbracket \cup \mathcal{S}^r \llbracket R_2 \rrbracket =$$

$$\mathcal{S}^r \llbracket R_1 + R_2 \rrbracket$$

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

$$\mathcal{S}^r \llbracket \text{dfn}(R_1^*) \rrbracket =$$

$$\mathcal{S}^r \llbracket ((R_1^1)^* (R_1^2)^* \dots (R_1^n)^*)^* \rrbracket =$$

(طبق نتیجه‌ای که از قضیه‌ی قبل گرفتیم)

$$\mathcal{S}^r \llbracket (R_1^1 + R_1^2 + \dots + R_1^n)^* \rrbracket =$$

$$\mathcal{S}^r \llbracket (R_1)^* \rrbracket =$$

$$\mathcal{S}^r \llbracket R_1^* \rrbracket$$

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

$$\mathcal{S}^r \llbracket \text{dfn}(R_1^+) \rrbracket =$$

$$\mathcal{S}^r \llbracket \text{dfn}(R_1 R_1^*) \rrbracket$$

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

$$\mathcal{S}^r \llbracket \text{dfn}(R_1 R_1^*) \rrbracket = \mathcal{S}^r \llbracket R_1 R_1^* \rrbracket = \mathcal{S}^r \llbracket R_1^+ \rrbracket$$

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

$$\mathcal{S}^r \llbracket \text{dnf}((R_1)) \rrbracket =$$

$$\mathcal{S}^r \llbracket \text{dnf}(R_1) \rrbracket =$$

(طبق فرض استقرا):

$$\mathcal{S}^r \llbracket R_1 \rrbracket =$$

$$\mathcal{S}^r \llbracket (R_1) \rrbracket$$

□

۳.۱.۳ سر و دم عبارات منظم

در این بخش، تابعی را روی عبارات منظم تعریف می‌کنیم که یک عبارت منظم را می‌گیرد و یک زوج از عبارات منظم را تحویل می‌دهد، سپس به بیان یک قضیه در مورد این تابع می‌پردازیم. این تابع را با fstnxt نشان می‌دهیم. قرار است این تابع یک عبارت منظم را بگیرد و آن را به این شکل تجزیه کند که اولین زوج 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 $\mathbb{R} \times \mathbb{R} : (\mathbb{R}^+ \cap \mathbb{R}^+) \rightarrow \text{fstnxt}$ به شکل زیر تعریف می‌کنیم:

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

$$\blacktriangleleft \text{fstnxt}(R_1 R_2) = \text{fstnxt}(R_2)$$

$$\text{where } R_1 \in \mathbb{R}_\varepsilon$$

$$\blacktriangleleft \text{fstnxt}(R_1 R_2) = \langle R_1^n \in \mathbb{R}_\varepsilon ? \langle R_1^f, R_2 \rangle : \langle R_1^f, R_1^n \bullet R_2 \rangle \rangle$$

$$\text{where } R_1 \notin \mathbb{R}_\varepsilon \text{ and } \text{fstnxt}(R_1) = \langle R_1^f, R_1^n \rangle$$

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

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

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

از این تعریف در صورتی از واری 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 گرامر برای $\mathbb{R}^+ \cap \mathbb{R}^\dagger$ می‌آوریم.

قضیه ۹.۳. گرامر زیر زبان $\mathbb{R}^+ \cap \mathbb{R}^\dagger$ را توصیف می‌کند.

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

$$R ::= L : B \mid \varepsilon R_2 \mid R_1 \varepsilon \mid R_1 R_2 \mid R_1^+ \mid (R_1)$$

اثبات. نام مجموعه‌ی عبارات منظم تولید شده با گرامر 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 گرامر بالا استفاده کنیم:

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

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

$$\blacktriangleright R = \varepsilon R_2 :$$

با فرض اینکه $R_2 \in \mathbb{R}^+ \cap \mathbb{R}^\dagger$ که فرض استقراست، طبق گرامر \mathbb{R}^+ داریم $\varepsilon R_2 \in \mathbb{R}^+$ و طبق گرامر \mathbb{R}^\dagger داریم چون $R_2 \in \mathbb{R}^\dagger$ و $\varepsilon \in \mathbb{R}^\dagger$ ، پس $\varepsilon R_2 \in \mathbb{R}^\dagger$. پس داریم $\varepsilon R_2 \in \mathbb{R}^+ \cap \mathbb{R}^\dagger$.

$$\blacktriangleright R = R_1 \varepsilon :$$

مشابه مورد قبل ثابت می شود.

$$\blacktriangleright R = R_1 R_2 :$$

طبق فرض استقرا داریم $R_1, R_2 \in \mathbb{R}^+ \cap \mathbb{R}^+$ و در هر دو گرامر عملگر چسباندن را داریم. پس این مورد هم اثبات می شود.

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

مثل مورد قبل، چون عملگر $^+$ در هر دو گرامر موجود است، به کمک فرض استقرا اثبات می شود.

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

مثل مورد قبل اثبات می شود.

در اینجا اثبات $\mathbb{R}' \subseteq \mathbb{R}^+ \cap \mathbb{R}^+$ کامل می شود. حال به سراغ اثبات $\mathbb{R}^+ \cap \mathbb{R}^+ \subseteq \mathbb{R}'$ می رویم. برای اثبات این بخش، با فرض اینکه $R \in \mathbb{R}^+ \cap \mathbb{R}^+$ ، از استقرا روی ساختار اعضای \mathbb{R}' استفاده می کنیم (این اثبات می تواند با استقرا روی ساختار اعضای \mathbb{R}^+ هم انجام شود).

$$\blacktriangleright R = \varepsilon :$$

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

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

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

$$\blacktriangleright R = R_1 R_2 :$$

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

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

چون به ازای هیچ عبارت منظم R_1 ی R_1^* داخل \mathbb{R}^+ نمی افتد، پس بررسی این مورد هم مورد نیاز نیست.

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

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

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

شبه به مورد 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 گرامر را در اینجا ارائه کرده‌ایم.

قضیه ۱۰.۳. برای هر عبارت منظم $R \in \mathbb{R}^+ \cap \mathbb{R}^{\dagger}$ اگر $R = \langle L : B, R' \rangle$ $\text{fstnxt}(R) = \text{fstnxt}(R)$ آنگاه $R' \in \mathbb{R}^+$ و $R \approx L : B \bullet R'$.

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

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

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

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

طبق تعریف تابع سر و دم، داریم $\text{fstnxt}(\varepsilon R_2) = \text{fstnxt}(R_2)$. فرض استقرا این است که اگر $\text{fstnxt}(R_2) = \langle L : B, R'_2 \rangle$ آنگاه $R'_2 \in \mathbb{R}^{\dagger}$ و $R_2 \approx L : B \bullet R'_2$. پس $\text{fstnxt}(R) = \langle L : B, R'_2 \rangle$ که همان طور که گفتیم طبق فرض استقرا $R'_2 \in \mathbb{R}^{\dagger}$ و از طرف دیگر:

$$R = \varepsilon R_2 \approx R_2 \approx L : B \bullet R'_2$$

$$\blacktriangleright R = R_1 \varepsilon;$$

در این حالت امکان ندارد $R_1 = \varepsilon$ باشد، چون در آن صورت خواهیم داشت، $R = \varepsilon \varepsilon \in \mathbb{R}_\varepsilon$ که تناقض است چون $\varepsilon \varepsilon$ در دامنه‌ی تابع سر و دم نیست. طبق تعریف سر و دم اگر داشته باشیم $\text{fstnxt}(R_1) = \langle L : B, R'_1 \rangle$ در این صورت بنا بر اینکه $R'_1 \in \mathbb{R}_\varepsilon$ برقرار باشد یا نباشد، دو حالت را داریم:

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

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

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

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

$$L : B \bullet R'_1 \bullet \varepsilon \approx R_1 \varepsilon = R$$

$$\blacktriangleright R = R_1 R_2;$$

اگر یکی از R_1 و R_2 برابر ε باشد، 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 $\text{fstnxt}(R_1) = \langle L : B, R'_1 \rangle$ ، مسئله بنا به اینکه $R'_1 \in \mathbb{R}_\varepsilon$ برقرار هست یا خیر، به دو حالت افراز می‌شود.

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

در این صورت $\text{fstnxt}(R) = \langle L : B, R'_1 \bullet R_2 \rangle$. مانند استدلال که در مورد قبلی آوردیم، خواهیم داشت $R'_1 \bullet R_2 \in \mathbb{R}^\dagger$. علاوه بر این، طبق فرض استقرا داریم $R'_1 \approx L : B \bullet R'_1$ پس:

$$L : B \bullet R'_1 \bullet R_2 \approx R_1 \bullet R_2 = R$$

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

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

در این صورت $\text{fstnxt}(R) = \langle L : B, R_2 \rangle$. مثل حالت های قبل ثابت می شود که $R_2 \in \mathbb{R}^\dagger$ و داریم:

$$R = L : B \bullet R_2 \Rightarrow L : B \bullet R_2 \approx R$$

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

با فرض اینکه $\text{fstnxt}(R_1) = \langle L : B, R'_1 \rangle$ بنا به اینکه $R'_1 \in \mathbb{R}_\varepsilon$ برقرار باشد یا نباشد، دو حالت خواهیم داشت:

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

در این صورت طبق تعریف تابع سر و دم، $\text{fstnxt}(R) = \langle L : B, R_1^* \rangle$ را داریم. چون داخل R'_1 و $L : B$ 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:

$$\text{fstnxt}(R_1) = \langle L : B, R'_1 \rangle \rightarrow R'_1 \in \mathbb{R}^\dagger \wedge L : B \bullet R'_1 \approx R_1$$

(عبارت بالا فرض استقراست).

$$R_1^* \approx (L : B \bullet R'_1)^* \approx (L : B \bullet \varepsilon)^* \approx (L : B)^*$$

(هم ارزی وسطی به خاطر این است که R'_1 عضو \mathbb{R}^\dagger است. اگر یکی از دو هم ارزی دیگر هم برقرار نباشند کلا عملگر * خوش تعریف نخواهد بود، پس این دو هم ارزی باید برقرار باشند.)

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

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

با توجه به تعریف تابع سر و دم و فرض استقرا که پیشتر بیان کرده ایم، در این حالت داریم $\text{fstnxt}(R) = \langle L : B, R'_1 \bullet R_1^* \rangle$. به همان دلیل مورد قبلی، می دانیم که R_1^* عضو \mathbb{R}^\dagger است و طبق فرض استقرا داریم $R'_1 \in \mathbb{R}^\dagger$. بنابراین داریم $R'_1 \bullet R_1^* \in \mathbb{R}^\dagger$.

با استفاده از فرض استقرا داریم:

$$L : B \bullet R'_1 \approx R_1$$

$$\Rightarrow L : B \bullet R'_1 \bullet R_1^* \approx R_1 R_1^* \approx R_1^+$$

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

□

از فرض استقرا نتیجه می شود.

این بخش، در این قسمت به پایان می رسد. حال ابزارهای کافی برای بیان روش واریسی مدل به شکل جدیدی که مد نظر است را داریم.

۲.۳ واریسی مدل منظم

همان طور که پیش تر گفتیم، می خواهیم در این فصل یک صورت معادل با صورتی که در فصل پیش برای روش واریسی مدل آورده شده بود را ارائه کنیم. تا اینجا این فصل، صرفاً به معرفی چند مفهوم که برای بیان صورت جدید به آن ها احتیاج داریم، پرداخته ایم. در این بخش ابتدا این صورت جدید را بیان می کنیم و سپس اثبات می کنیم که صورت جدید با صورت قبلی معادل است. همان طور که پیش تر هم اشاره شد، تفاوت این بیان با بیان قبلی این است که این بیان روی ساختار عبارات منظم تعریف شده است، در حالیکه، صورت قبلی ساختاری نداشت.

۱.۲.۳ صورت

در نهایت، برای تعریف صورت به یک تابع به نام M خواهیم رسید که در ورودی اش، یک زوج متشکل از یک محیط اولیه و یک عبارت منظم را در کنار یک برنامه می گیرد و در خروجی، همه ی ردهای پیشوندی موجود در معنای برنامه را که با عبارت منظم سازگار هستند، داخل یک مجموعه بر می گرداند. اما در این بین، مفهوم سازگاری یک رد پیشوندی با یک عبارت منظم چگونه مشخص می شود؟ این نکته ای است که تا به حال در مورد آن بحث نکرده ایم و حالا می خواهیم، تعریف تابع 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 می گوید 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. (وارسی‌گرد پیشوندی): به تابع \mathcal{M}^t از نوع $(\mathbb{B} \times \mathbb{R}^{\dagger}) \rightarrow \mathbb{G}^{+\infty} \rightarrow (\mathbb{E}\mathbb{V} \times \mathbb{R}^{\dagger})$ واریسی‌گرد 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}_ε هم ضابطه‌ی بالایی برقرار است. دو ضابطه‌ی پایینی برای عبارات منظم عضو $\mathbb{R}^+ \cap \mathbb{R}^{\dagger}$ هستند.)

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

$$\blacktriangleleft \mathcal{M}^t \langle \underline{\rho}, R \rangle \pi = \langle \langle \underline{\rho}, \langle l_1, \rho_1 \rangle \rangle \in \mathcal{S}^r \llbracket L : B \rrbracket ? \mathcal{M}^t \langle \underline{\rho}, R' \rangle \pi' : \langle F, R \rangle \rangle$$

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

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 عبارات منظم انجام دهد، این تابع روی عبارات منظمی که + ندارند انجام می‌دهد.

تعریف ۱۲.۳. (وارسی مدل منظم محدود به \mathbb{R}^\dagger): به تابع \mathcal{M}^\dagger از نوع $(\mathbb{E}\mathbb{V} \times \mathbb{R}^{\dagger}) \rightarrow P(\mathbb{G}^{+\infty}) \rightarrow P(\mathbb{G}^{+\infty} \times \mathbb{R}^{\dagger})$ می‌گوییم واریسی مدل منظم محدود به \mathbb{R}^\dagger . ضابطه‌ی این تابع به شکل زیر است:

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

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، صرفاً 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 تطابقش 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 $(\mathbb{E}\mathbb{V} \times P(\mathfrak{G}^{+\infty})) \rightarrow P(\mathfrak{G}^{+\infty}) \rightarrow (\mathbb{E}\mathbb{V} \times P(\mathfrak{G}^{+\infty}))$ واریسی مدل منظم می‌گوییم که ضابطه‌ی زیر را دارد:

$$\mathcal{M}\langle \underline{\rho}, R \rangle \Pi = \bigcup_{i=1}^n \{ \langle \underline{\rho}, \pi \rangle \mid \exists R' \in \mathbb{R} : \langle \pi, R' \rangle \in \mathcal{M}^t \langle \underline{\rho}, R_i \rangle \Pi \}$$

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

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

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

و برقرار بودن این رابطه با شرط زیر تعریف می‌شود:

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

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

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

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

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

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

که طبق تعریف \mathcal{M}^t یعنی:

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

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

□

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 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 زبانی که تعریف کرده‌ایم غیرقطعی باشد. در صورتیکه، در زبان و معناشناسی این زبان 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 $M(\rho, R)S^*[P]$ در نظر بگیریم. می‌توانیم هر دسته از این افراز را متناظر با دسته‌ای در افرازی که روی $S^*[P]$ 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 $S^*[P]$ ، رد پیشوندی ماکسیمال این دسته در دسته‌ی متناظر در $M(\rho, R)S^*[P]$ وجود داشته باشد، در این صورت پس حتماً $M(\rho, R)S^*[P] = \{\rho\} \times S^*[P]$. اگر دسته‌ای از افراز روی $M(\rho, R)S^*[P]$ دارای عضو ماکسیمال متفاوتی نسبت به همان دسته روی $S^*[P]$ باشد، قطعاً این عضو ماکسیمال کوتاه‌تر از عضو ماکسیمال همان دسته در $S^*[P]$ است و این یعنی ناسازگاری‌ای با عبارت منظم مورد بررسی وجود داشته است. محل وقوع ناسازگاری را تابع M^l می‌تواند به ما بگوید. محل ناسازگاری عبارت منظمی است که زوج عضو ماکسیمال دسته‌ی مورد نظر در خروجی یکی از اعمال های M^l روی بخش‌های مختلف $\text{dnf}(R)$ 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 و تمامیت 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 ρ یک محیط اولیه باشند، آنگاه داریم:

$$P, \rho \models_r R \Rightarrow P, \rho \models R$$

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

$$\{\rho\} \times S^*[P] \subseteq \mathcal{M}(\rho, R) S^*[P]$$

ثابت کنیم:

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

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

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

در مورد عبارت منظم R ، فرض می‌کنیم $\text{dnf}(R) = R_1 + R_2 + \dots + R_n$ که هر R_i ذکر شده عضو \mathbb{R}^t است. فرض می‌کنیم $\langle \rho, \pi \rangle \in \mathcal{M}(\rho, R) S^*[P]$ آنگاه وجود دارد k و R' که $\langle \pi, R' \rangle \in \mathcal{M}^t(\rho, R_k) S^*[P]$ که طبق تعریف \mathcal{M}^t خواهیم داشت:

$$\pi \in S^*[P] \wedge \mathcal{M}^t(\rho, R_k) \pi = \langle T, R' \rangle$$

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

$$\mathcal{M}^t(\rho, R_k) \pi = \langle T, R' \rangle$$

در مورد R_k دو حالت داریم، یا $R_k \approx \varepsilon$ برقرار است، یا اینگونه نیست ($R_k \not\approx \varepsilon$).

$$\blacktriangleright R_k \approx \varepsilon :$$

در این صورت می‌توانیم، ثابت کنیم:

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

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

$$\begin{aligned} R \bullet (? : T)^* &\approx (R_1 + R_2 + \dots + R_m) \bullet (? : T)^* \\ &\approx R_1 \bullet (? : T)^* + R_2 \bullet (? : T)^* + \dots + R_n \bullet (? : T)^* \end{aligned}$$

چون $R_k \approx \varepsilon$ ، داریم:

$$\begin{aligned} &R_1 \bullet (? : T)^* + R_2 \bullet (? : T)^* + \dots + R_k \bullet (? : T)^* + \dots + R_n \bullet (? : T)^* \\ &\approx R_1 \bullet (? : T)^* + R_2 \bullet (? : T)^* + \dots + \varepsilon \bullet (? : T)^* + \dots + R_n \bullet (? : T)^* \end{aligned}$$

و از طرف دیگر داریم:

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

پس $\text{prefix}(\mathcal{S}^r[\llbracket R \bullet (? : T)^* \rrbracket])$ 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:

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

که این گزاره نتیجه می‌دهد:

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

► $R_k \not\approx \varepsilon$:

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

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

و تعریف می‌کنیم:

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

داریم:

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

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

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

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

$$\mathcal{M}^t \langle \underline{\rho}, R_k^1 \rangle \pi(1) = \mathcal{M}^t \langle \underline{\rho}, R_k^2 \rangle \pi(2) \quad \text{where } \text{fstnxt}(R_k^1) = \langle L_k^2 : B_k^2, R_k^2 \rangle$$

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

$$\mathcal{M}^t \langle \underline{\rho}, R_k^{h-1} \rangle \pi(h-1) = \mathcal{M}^t \langle \underline{\rho}, R_k^h \rangle \pi(h) \quad \text{where } \text{fstnxt}(R_k^{h-1}) = \langle L_k^h : B_k^h, R_k^h \rangle$$

در حالیکه، $\varepsilon \approx R_k^h$. اگر گزاره‌ی $\varepsilon \approx R_k^h$ 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:

$$R_k \approx L_k^1 : B_k^1 \bullet L_k^2 : B_k^2 \bullet \dots \bullet L_k^h : B_k^h$$

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:

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

که این یعنی داریم:

$$\forall j : 1 \leq j \leq h \rightarrow \langle \underline{\rho}, \langle l_j, \rho_j \rangle \rangle \in \mathcal{S}^r \llbracket L_j : B_j \rrbracket$$

$$\Rightarrow \langle \underline{\rho}, \pi \rangle \in \mathcal{S}^r[\llbracket R_k \bullet (? : T)^* \rrbracket] \Rightarrow \langle \underline{\rho}, \pi \rangle \in \text{prefix}(\mathcal{S}^r[\llbracket R \bullet (? : T)^* \rrbracket])$$

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

در این صورت داریم:

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

که یعنی:

$$\forall j : 1 \leq j \leq l \rightarrow \langle \underline{\rho}, \langle l_j, \rho_j \rangle \rangle \in \mathcal{S}^r[\llbracket L_j : B_j \rrbracket]$$

$$\Rightarrow \langle \underline{\rho}, \pi \rangle \in \text{prefix}(\mathcal{S}^r[\llbracket R_k \rrbracket]) \Rightarrow \langle \underline{\rho}, \pi \rangle \in \text{prefix}(\mathcal{S}^r[\llbracket R \bullet (? : T)^* \rrbracket])$$

پس در کل می‌توانیم، بگوییم

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

□

و اثبات قضیه تمام می‌شود.

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:

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

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

$$\{\underline{\rho}\} \times \mathcal{S}^*[\llbracket P \rrbracket] \not\subseteq \mathcal{M} \langle \underline{\rho}, R \rangle \mathcal{S}^*[\llbracket P \rrbracket] \Rightarrow \exists \pi : \langle \underline{\rho}, \pi \rangle \in \{\underline{\rho}\} \times \mathcal{S}^*[\llbracket P \rrbracket] \wedge \langle \underline{\rho}, \pi \rangle \notin \mathcal{M} \langle \underline{\rho}, R \rangle \mathcal{S}^*[\llbracket P \rrbracket]$$

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

$$R_i \approx L_i^1 : B_i^1 \bullet L_i^2 : B_i^2 \bullet \dots \bullet L_i^n : B_i^n$$

و

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

می‌توانیم، در ادامه‌ی فرض خلف، نتیجه بگیریم:

$$\forall i : 1 \leq i \leq n \rightarrow \langle \underline{\rho}, \pi \rangle \notin \mathcal{M}^t \langle \underline{\rho}, R_i \rangle \mathcal{S}^* \llbracket P \rrbracket$$

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

در این صورت، خواهیم داشت:

$$\exists j : \langle \underline{\rho}, \langle l_j, \rho_j \rangle \rangle \notin \mathcal{S}^r \llbracket L_i^j : B_i^j \rrbracket \Rightarrow \langle \underline{\rho}, \pi \rangle \notin \mathcal{S}^r \llbracket R_i \bullet (? : T)^* \rrbracket$$

از نتیجه‌ی آخر می‌توانیم ثابت کنیم $\langle \underline{\rho}, \pi \rangle \notin \text{prefix}(\mathcal{S}^r \llbracket R_i \bullet (? : T)^* \rrbracket)$. چون اگر غیر از این باشد، یعنی اگر فرض کنیم $\langle \underline{\rho}, \pi' \rangle$ عضو $\text{prefix}(\mathcal{S}^r \llbracket R_i \bullet (? : T)^* \rrbracket)$ هست، اما عضو $\mathcal{S}^r \llbracket R_i \bullet (? : T)^* \rrbracket$ 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 عبارت بالایی که دارای سور وجودی است)، پس باز هم $\pi \neq \pi'$. توجه شود که $\langle \underline{\rho}, \pi \rangle \notin \mathcal{S}^r \llbracket R_i \bullet (? : T)^* \rrbracket$ 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 واریسی مدل اضافه کردیم و حالا می‌خواهیم، ساختار زبانمان را به کار اضافه کنیم. این آخرین تلاش [۶] برای گسترش کار بوده است. یعنی واریسی مدل به شکل جدید تعریف شده است و معادل بودن آن با صورت قبلی واریسی 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 مورد تعریف بیان شده است. بنابراین، از ارائه‌ی دوباره‌ی این جزئیات خودداری کرده‌ایم.

تعریف ۱.۴. تابع \hat{M} را از نوع $(\mathbb{V} \times P(\mathcal{G}^{+\infty})) \rightarrow P(\mathcal{G}^{+\infty}) \rightarrow (\mathbb{V} \times \mathbb{R})$ واریسی مدل ساختارمند می‌نامیم (ضابطه‌ی تابع در ادامه‌ی متن آمده است).

در ادامه، ممکن است به جای $\hat{M}(\underline{\rho}, R)S^*[P]$ از $\hat{M}(\underline{\rho}, R)[P]$ استفاده کرده باشیم، یعنی در اشاره به تابع S^* به براکت‌ها $[[\]]$ قناعت کرده باشیم.

تعریف روی ساختار مجموعه‌ی $P \cup S \cup S$ انجام شده است. تقریباً کاری شبیه به اثبات لمی که در بحث تصمیم‌ناپذیری در فصل سوم داشتیم. در ادامه، قسمت‌های مختلف تعریف \hat{M} را به ازای برنامه‌ی P ، محیط اولیه‌ی $\underline{\rho}$ و عبارت منظم R تعریف می‌کنیم. یعنی در حال تعریف $\hat{M}(\underline{\rho}, R)S^*[P]$ هستیم، روی ساختار برنامه‌ها یعنی P .

$$\blacktriangleleft P = SI :$$

$$\hat{\mathcal{M}}\langle \underline{\rho}, R \rangle \mathcal{S}^* \llbracket P \rrbracket = \bigcup_{i=1}^n \{ \langle \underline{\rho}, \pi \rangle \mid \exists R' \in \mathbb{R}, \langle \pi, R' \rangle \hat{\mathcal{M}}\langle \underline{\rho}, R_i \rangle \mathcal{S}^* \llbracket SI \rrbracket \}$$

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

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

در ادامه با توجه به تعریف قبل، به بیان تعریف $\hat{\mathcal{M}}\langle \underline{\rho}, R \rangle$ 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 $\hat{\mathcal{M}}\langle \underline{\rho}, R \rangle$ در عمل روی مجموعه‌ی معنای برنامه‌ها تعریف می‌شود. مثلاً، به ازای $\Pi \subseteq \mathbb{S}^{+\infty}$ 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 اصلاً به ازای چنین ورودی‌ای خروجی ندارد. به عبارت دیگر، تابع جزئی است. مشابه $\mathcal{M}\langle \underline{\rho}, R \rangle$ خروجی $\hat{\mathcal{M}}\langle \underline{\rho}, R \rangle$ 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.

$$\blacktriangleleft \hat{\mathcal{M}}\langle \underline{\rho}, \varepsilon \rangle \llbracket S \rrbracket = \{ \langle \pi, \varepsilon \rangle \mid \pi \in \mathcal{S}^* \llbracket S \rrbracket \}$$

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

$$\blacktriangleleft \hat{\mathcal{M}}\langle \underline{\rho}, R \rangle \llbracket SI \rrbracket = \hat{\mathcal{M}}\langle \underline{\rho}, R \rangle \llbracket SI' \rrbracket \cup$$

$$\{ \langle \pi \langle \text{at} \llbracket S \rrbracket, \rho \rangle \pi', R'' \rangle \mid \langle \pi \langle \text{at} \llbracket S \rrbracket, \rho \rangle, R' \rangle \in \hat{\mathcal{M}}\langle \underline{\rho}, R \rangle \llbracket SI' \rrbracket \wedge$$

$$\langle \langle \text{at} \llbracket S \rrbracket, \rho \rangle \pi', R'' \rangle \in \hat{\mathcal{M}}\langle \underline{\rho}, R' \rangle \llbracket S \rrbracket \}$$

از اینجا به بعد، با تعاریف طولی‌تری از آنچه تا حالا داشتیم، روبرو هستیم. هرچند که مفهوم چندان پیچیده‌ای پشت این تعاریف نیست. به طور خلاصه، تعریف بالا می‌گوید، تعریف $\hat{\mathcal{M}}\langle \underline{\rho}, R \rangle \llbracket SI \rrbracket$ وابسته به تعریف $\hat{\mathcal{M}}\langle \underline{\rho}, R \rangle \llbracket S \rrbracket$ و $\hat{\mathcal{M}}\langle \underline{\rho}, R \rangle \llbracket SI' \rrbracket$ است. ردهای پیشوندی‌ای که داخل این دو مجموعه هستند، به یکدیگر چسبانده می‌وند، به طوریکه اول ردهای داخل $\hat{\mathcal{M}}\langle \underline{\rho}, R \rangle \llbracket SI' \rrbracket$ قرار می‌گیرند و بعد ردهای داخل $\hat{\mathcal{M}}\langle \underline{\rho}, R \rangle \llbracket S \rrbracket$. ردهای داخل $\hat{\mathcal{M}}\langle \underline{\rho}, R \rangle \llbracket SI' \rrbracket$ به تنهایی نیز داخل

$\hat{\mathcal{M}}^\dagger \langle \underline{\rho}, R \rangle \llbracket SI \rrbracket$ قرار می‌گیرند. این تعریف بر اساس تعریف $S^* \llbracket SI \rrbracket$ ارائه شده است. اساس گرفتن تعریف تابع 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 $SI = \epsilon$ و $R \in \mathbb{R}^\dagger \cap \mathbb{R}^+$ داریم:

$$\begin{aligned} \blacktriangleleft \hat{\mathcal{M}}^\dagger \langle \underline{\rho}, R \rangle \llbracket SI \rrbracket = \\ \{ \langle \langle at \llbracket SI \rrbracket, \rho \rangle, R' \rangle \mid \langle \underline{\rho}, \langle at \llbracket SI \rrbracket, \rho \rangle \rangle \in \mathcal{S}^r \llbracket L : B \rrbracket \} \\ \text{where } \text{fstnxt}(R) = \langle L : B, R' \rangle \end{aligned}$$

یعنی خروجی تابع به ازای این ورودی مجموعه‌ای است، شامل همه‌ی ردهای پیشوندی تک عضوی‌ای که محیط آن‌ها اولین سر عبارت منظم $(L : B)$ 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 $S = x \doteq A$ و $R \in \mathbb{R}^\dagger \cap \mathbb{R}^+$ داریم:

$$\begin{aligned} \blacktriangleleft \hat{\mathcal{M}}^\dagger \langle \underline{\rho}, R \rangle \llbracket S \rrbracket = \\ \{ \langle \langle at \llbracket S \rrbracket, \rho \rangle, R' \rangle \mid \langle \underline{\rho}, \langle at \llbracket S \rrbracket, \rho \rangle \rangle \in \mathcal{S}^r \llbracket L : B \rrbracket \} \\ \cup \{ \langle \langle at \llbracket S \rrbracket, \rho \rangle \langle aft \llbracket S \rrbracket, \rho[x \leftarrow \mathcal{A}[A]\rho] \rangle, \epsilon \rangle \mid R' \in \mathbb{R}_\epsilon \wedge \\ \langle \underline{\rho}, \langle at \llbracket S \rrbracket, \rho \rangle \rangle \in \mathcal{S}^r \llbracket L : B \rrbracket \} \\ \cup \{ \langle \langle at \llbracket S \rrbracket, \rho \rangle \langle aft \llbracket S \rrbracket, \rho[x \leftarrow \llbracket A \rrbracket \rho] \rangle, R'' \rangle \mid R' \notin \mathbb{R}_\epsilon \wedge \\ \langle \underline{\rho}, \langle at \llbracket S \rrbracket, \rho \rangle \rangle \in \mathcal{S}^r \llbracket L : B \rrbracket \wedge \langle L' : B', R'' \rangle = \text{fstnxt}(R') \wedge \\ \langle \underline{\rho}, \langle aft \llbracket S \rrbracket, \rho[x \leftarrow \mathcal{A}[A]\rho] \rangle \rangle \in \mathcal{S}^r \llbracket L' : B' \rrbracket \} \\ \text{where } \text{fstnxt}(R) = \langle L : B, R' \rangle \end{aligned}$$

این تابع عبارت دستوری را به همراه زوجی 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 $R \in \mathbb{R}^{\dagger} \cup \mathbb{R}^+$ و S_t if (B) $S =$ داریم:

$$\begin{aligned}
\blacktriangleleft \hat{\mathcal{M}}^{\dagger}(\underline{\rho}, R)[S] = & \\
& \{ \langle \langle at[S], \rho \rangle, R' \rangle | \langle \underline{\rho}, \langle at[S], \rho \rangle \rangle \in \mathcal{S}^r[L' : B'] \} \\
& \cup \{ \langle \langle at[S], \rho \rangle \langle at[S_t], \rho \rangle \pi, R'' | \mathcal{B}[B]\rho = T \wedge \\
& \quad \langle \underline{\rho}, \langle at[S], \rho \rangle \rangle \in \mathcal{S}^r[L' : B'] \wedge \\
& \quad \langle \langle at[S_t], \rho \rangle \pi, R'' \rangle \in \hat{\mathcal{M}}^{\dagger}(\underline{\rho}, R')[S_t] \} \\
& \cup \{ \langle \langle at[S], \rho \rangle \langle aft[S], \rho \rangle, \varepsilon \rangle | \mathcal{B}[B]\rho = F \wedge R' \in \mathbb{R}_{\varepsilon} \wedge \\
& \quad \langle \underline{\rho}, \langle [S], \rho \rangle \rangle \in \mathcal{S}^r[L' : B'] \} \\
& \cup \{ \langle \langle at[S], \rho \rangle \langle aft[S], \rho \rangle, R'' \rangle | \mathcal{B}[B]\rho = F \wedge R' \notin \mathbb{R}_{\varepsilon} \wedge \\
& \quad \langle \underline{\rho}, \langle at[S], \rho \rangle \rangle \in \mathcal{S}^r[L' : B'] \wedge \langle L'' : B'', R'' \rangle = \text{fstnxt}(R') \wedge \\
& \quad \langle \underline{\rho}, \langle aft[S], \rho \rangle \rangle \in \mathcal{S}^r[L'' : B''] \} \\
& \text{where } \text{fstnxt}(R) = \langle L' : B', R' \rangle
\end{aligned}$$

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

$$\begin{aligned}
\blacktriangleleft \hat{\mathcal{M}}^{\dagger}(\underline{\rho}, R)[S] = & \\
& \{ \langle \langle at[S], \rho \rangle, R' \rangle | \langle \underline{\rho}, \langle at[S], \rho \rangle \rangle \in \mathcal{S}^r[L' : B'] \} \\
& \cup \{ \langle \langle at[S], \rho \rangle \langle at[S_t], \rho \rangle \pi, R'' | \mathcal{B}[B]\rho = T \wedge \\
& \quad \langle \underline{\rho}, \langle at[S], \rho \rangle \rangle \in \mathcal{S}^r[L' : B'] \wedge \\
& \quad \langle \langle at[S_t], \rho \rangle \pi, R'' \rangle \in \hat{\mathcal{M}}^{\dagger}(\underline{\rho}, R')[S_t] \} \\
& \cup \{ \langle \langle at[S], \rho \rangle \langle at[S_f], \rho \rangle \pi, R'' | \mathcal{B}[B]\rho = F \wedge
\end{aligned}$$

$$\begin{aligned}
& \langle \underline{\rho}, \langle at[S], \rho \rangle \rangle \in \mathcal{S}^r[L' : B'] \wedge \\
& \langle \langle at[S_f], \rho \rangle \pi, R'' \rangle \in \hat{\mathcal{M}}^\dagger \langle \underline{\rho}, R' \rangle [S_f] \} \\
& \text{where } fstnxt(R) = \langle L' : B', R' \rangle
\end{aligned}$$

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 $\hat{\mathcal{M}}^\dagger \langle \underline{\rho}, R \rangle [S]$ 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 $\hat{\mathcal{M}}^\dagger \langle \underline{\rho}, R' \rangle [S_t]$ (که R' نیز دم R است) حضور دارد یا خیر. اگر عبارت بولی در محیط مذکور ارزش غلط داشته باشد، حضور رد پیشوندی در $\hat{\mathcal{M}}^\dagger \langle \underline{\rho}, R \rangle [S]$ 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 $R \in \mathbb{R}^+ \cup \mathbb{R}^+$ و $S = \text{break}$; داریم:

$$\begin{aligned}
& \blacktriangleleft \hat{\mathcal{M}}^\dagger \langle \underline{\rho}, R \rangle [S] = \\
& \{ \langle \langle at[S], \rho \rangle, R' \rangle | \langle \underline{\rho}, \langle at[S], \rho \rangle \rangle \in \mathcal{S}^r[L : B] \} \\
& \cup \{ \langle \langle at[S], \rho \rangle \langle brk - to[S], \rho \rangle, \varepsilon \rangle | R' \in \mathbb{R}_\varepsilon \wedge \\
& \quad \langle \underline{\rho}, \langle at[S], \rho \rangle \rangle \in \mathcal{S}^r[L : B] \} \\
& \cup \{ \langle \langle at[S], \rho \rangle \langle brk - to[S], \rho \rangle, R'' \rangle | R' \notin \mathbb{R}_\varepsilon \wedge \\
& \quad \langle \underline{\rho}, \langle at[S], \rho \rangle \rangle \in \mathcal{S}^r[L : B] \wedge \langle L' : B', R'' \rangle = fstnxt(R') \wedge \\
& \quad \langle \underline{\rho}, \langle brk - to[S], \rho \rangle \rangle \in \mathcal{S}^r[L' : B'] \} \\
& \text{where } fstnxt(R) = \langle L' : B', R' \rangle
\end{aligned}$$

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

$$\blacktriangleleft \hat{\mathcal{M}}^{\dagger}(\underline{\rho}, R) \llbracket S \rrbracket = lfp^{\subseteq} (\hat{\mathcal{F}}^{\dagger}(\underline{\rho}, R) \llbracket S \rrbracket)$$

$$\text{where } \hat{\mathcal{F}}^{\dagger}(\underline{\rho}, R)X = \{ \langle \langle at \llbracket S \rrbracket, \rho \rangle, R' \rangle \mid \rho \in \mathbb{E}V \wedge \langle \underline{\rho}, \langle at \llbracket S \rrbracket, \rho \rangle \rangle \in \mathcal{S}^r \llbracket L' : B' \rrbracket \}$$

$$\cup \{ \langle \pi_2 \langle at \llbracket S \rrbracket, \rho \rangle \langle aft \llbracket S \rrbracket, \rho \rangle, \varepsilon \rangle \mid \langle \pi_2 \langle at \llbracket S \rrbracket, \rho \rangle, \varepsilon \rangle \in X \wedge$$

$$\mathcal{B} \llbracket B \rrbracket \rho = F \}$$

$$\cup \{ \langle \pi_2 \langle at \llbracket S \rrbracket, \rho \rangle \langle aft \llbracket S \rrbracket, \rho \rangle, \varepsilon \rangle \mid \langle \pi_2 \langle at \llbracket S \rrbracket, \rho \rangle, R'' \rangle \in X \wedge$$

$$\mathcal{B} \llbracket B \rrbracket \rho = F \wedge R'' \notin \mathbb{R}_{\varepsilon} \wedge \langle L' : B', R' \rangle = \text{fstnxt}(R'') \wedge R' \in \mathbb{R}_{\varepsilon} \wedge$$

$$\langle \underline{\rho}, \langle at \llbracket S \rrbracket, \rho \rangle \rangle \in \mathcal{S}^r \llbracket L' : B' \rrbracket \}$$

$$\cup \{ \langle \pi_2 \langle at \llbracket S \rrbracket, \rho \rangle \langle aft \llbracket S \rrbracket, \rho \rangle, R' \rangle \mid \langle \pi_2 \langle at \llbracket S \rrbracket, \rho \rangle, R'' \rangle \in X \wedge$$

$$\mathcal{B} \llbracket B \rrbracket \rho = F \wedge R'' \notin \mathbb{R}_{\varepsilon} \wedge \langle L' : B', R' \rangle = \text{fstnxt}(R'') \wedge R''' \notin \mathbb{R}_{\varepsilon} \wedge$$

$$\langle \underline{\rho}, \langle at \llbracket S \rrbracket, \rho \rangle \rangle \in \mathcal{S}^r \llbracket L' : B' \rrbracket \wedge \langle L'' : B'', R' \rangle = \text{fstnxt}(R''') \wedge$$

$$\langle \underline{\rho}, \langle aft \llbracket S \rrbracket, \rho \rangle \rangle \in \mathcal{S}^r \llbracket L'' : B'' \rrbracket \}$$

$$\cup \{ \langle \pi_2 \langle at \llbracket S \rrbracket, \rho \rangle \langle at \llbracket S_b \rrbracket, \rho \rangle \pi_3, \varepsilon \rangle \mid \langle \pi_2 \langle at \llbracket S \rrbracket, \rho \rangle, \varepsilon \rangle \in X \wedge$$

$$\mathcal{B} \llbracket B \rrbracket \rho = T \wedge \langle at \llbracket S_b \rrbracket, \rho \rangle \pi_3 \in \mathcal{S}^* \llbracket S_b \rrbracket \}$$

$$\cup \{ \langle \pi_2 \langle at \llbracket S \rrbracket, \rho \rangle \langle at \llbracket S_b \rrbracket, \rho \rangle \pi_3, \varepsilon \rangle \mid \langle \pi_2 \langle at \llbracket S \rrbracket, \rho \rangle, R'' \rangle \in X \wedge$$

$$\mathcal{B} \llbracket B \rrbracket \rho = T \wedge R'' \notin \mathbb{R}_{\varepsilon} \wedge \langle L : B, \varepsilon \rangle = \text{fstnxt}(R'') \wedge$$

$$\langle \underline{\rho}, \langle at \llbracket S \rrbracket, \rho \rangle \rangle \in \mathcal{S}^r \llbracket L : B \rrbracket \wedge \langle at \llbracket S_b \rrbracket, \rho \rangle \pi_3 \in \mathcal{S}^* \llbracket S_b \rrbracket \}$$

$$\cup \{ \langle \pi_2 \langle at \llbracket S \rrbracket, \rho \rangle \langle at \llbracket S_b \rrbracket, \rho \rangle \pi_3, R' \rangle \mid \langle \pi_2 \langle at \llbracket S \rrbracket, \rho \rangle, R'' \rangle \in X \wedge$$

$$\mathcal{B} \llbracket B \rrbracket \rho = T \wedge R'' \notin \mathbb{R}_{\varepsilon} \wedge \langle L : B, R''' \rangle = \text{fstnxt}(R'') \wedge$$

$$\langle \underline{\rho}, \langle at \llbracket S \rrbracket, \rho \rangle \rangle \in \mathcal{S}^r \llbracket L : B \rrbracket \wedge R'''' \notin \mathbb{R}_{\varepsilon} \wedge$$

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

$$\langle \langle at[S_b], \rho \rangle \pi_3, R' \rangle \in \hat{\mathcal{M}}^\dagger \langle \underline{\rho}, R''' \rangle \llbracket S_b \rrbracket \}$$

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

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، واریسی مدل به صورت کوچک‌ترین نقطه ثابت 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 واریسی مدل اضافه انتظار داریم که عملگر نقطه ثابت 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 $R \in \mathbb{R}^\dagger \cup \mathbb{R}^+$ و $S = ;$ داریم:

$$\blacktriangleleft \hat{\mathcal{M}}^\dagger \langle \underline{\rho}, R \rangle \llbracket S \rrbracket = \{ \langle \langle at[S], \rho \rangle, R' \rangle \mid \langle \underline{\rho}, \langle at[S], \rho \rangle \rangle \in \mathcal{S}^r \llbracket L : B \rrbracket \}$$

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

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

$$\blacktriangleleft \hat{\mathcal{M}}^\dagger \langle \underline{\rho}, R \rangle \llbracket \{SI\} \rrbracket = \hat{\mathcal{M}}^\dagger \langle \underline{\rho}, R \rangle \llbracket SI \rrbracket$$

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

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

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

$$P, \underline{\rho} \models_s R \iff \{ \underline{\rho} \} \times \mathcal{S}^* \llbracket P \rrbracket \subseteq \hat{\mathcal{M}} \langle \underline{\rho}, R \rangle \mathcal{S}^* \llbracket P \rrbracket$$

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

فصل ۵

نتیجه گیری

واژه‌نامه فارسی به انگلیسی

ا

| | |
|----------------------------|-------------------|
| Execution | اجرا |
| Sheffer Operator | ادات شفر |
| Classical Connective | ادات کلاسیک |
| Modal Connective | ادات وجهی |
| Satisfiable | ارضایپذیر |
| Scheduling Algorithm | الگوریتم زمانبندی |

ب

| | |
|------------------------|------------------|
| Recursive | بازگشتی |
| Label | برچسب |
| Computer Program | برنامه کامپیوتری |

پ

| | |
|----------------------|------------|
| Python | پایتون |
| Processor | پردازنده |
| Implementation | پیاده‌سازی |

ت

| | |
|----------------|------|
| Function | تابع |
| Empty | تهی |

ج

| | |
|------------|------|
| Java | جاوا |
|------------|------|

چ

| | |
|---------------------|---------|
| Concatenation | چسباندن |
|---------------------|---------|

ح

| | |
|--------------|-------|
| Memory | حافظه |
| Case | حالت |
| Loop | حلقه |

خ

| | |
|--------------------|-----------|
| Property | ویژگی |
| Well-defined | خوش تعریف |

د

| | |
|--------------------|------------|
| True | درست |
| Truth | درستی |
| Verification | درستی یابی |

Sequence دنباله

ر

Prefix Trace رد پیشوندی

Formal Method روش صوری

ز

Imperative Programming Langugae زبان برنامه نویسی دستوری

Natural Language زبان طبیعی

Pair زوج مرتب

Subset زیرمجموعه

س

Structure ساختار

ش

Conditional شرط

Arrow شرطی

Countably Infinite شمارای نامتناهی

ص

Formal صوری

Formalization صوری‌گری

ع

| | |
|-----------------------------|---------------|
| Expression | عبارت |
| Boolean Expression | عبارت بولی |
| Arithmetic Expression | عبارت حسابی |
| Statement | عبارت دستوری |
| Positive Integers | عدد صحیح مثبت |
| Negative Integer | عدد صحیح منفی |
| Natural Number | عدد طبیعی |
| Member | عضو |
| Operator | عملگر |
| Boolean Operator | عملگر بولی |

غ

| | |
|-------------------------|----------|
| False | غلط |
| Non-deterministic | غیر قطعی |

ف

| | |
|----------------------|------------|
| Formula | فرمول |
| Atomic Formula | فرمول اتمی |
| Disjunction | فصل |

ق

| | |
|--------------------|-----------|
| Expressible | قابل بیان |
| Expressivity | قدرت بیان |

ک

Least Fixpoint کوچکترین نقطه ثابت

گ

Workflow گردش کار

ل

Statement List لیست عبارت‌های دستوری

م

Turing Machine ماشین تورینگ

Finite State Machine ماشین حالات منتهی

Register Machine ماشین رجیستر

Variable متغیر

Finite منتهی

Set مجموعه

Environment محیط

Model مدل

Model of Computation مدل محاسبه

Valid معتبر

Architecture معماری

Semantics معناشناسی

Prefix Trace Semantics معناشناسی رد پیشوندی

Formal Semantics معناشناسی صوری

Value مقدار

| | |
|-------------------------------------|----------------------|
| Boolean Value | مقدار بولی |
| Logic | منطق |
| Temporal Logic | منطق زمانی |
| Linear Temporal Logic | منطق زمانی خطی |
| Classical Logic | منطق کلاسیک |
| Classical Propositional Logic | منطق گزاره‌ای کلاسیک |
| Modal Logic | منطق موجهات |
| Subtraction | منها |

ن

| | |
|----------------|-----------|
| Infinite | نامتناهی |
| Syntax | نحو |
| Fixpoint | نقطه ثابت |

و

| | |
|----------------------|-----------|
| Model Checking | وارسی مدل |
| State | وضعیت |

ه

| | |
|------------------|--------|
| Equivalent | هم‌ارز |
|------------------|--------|

واژه‌نامه انگلیسی به فارسی

A

| | |
|-----------------------|-------------|
| Architecture | معماری |
| Arithmetic Expression | عبارت حسابی |
| Arrow | شرطی |
| Atomic Formula | فرمول اتمی |

B

| | |
|--------------------|------------|
| Boolean Expression | عبارت بولی |
| Boolean Operator | عملگر بولی |
| Boolean Value | مقدار بولی |

C

| | |
|-------------------------------|----------------------|
| Case | حالت |
| Classical Connective | ادات کلاسیک |
| Classical Logic | منطق کلاسیک |
| Classical Propositional Logic | منطق گزاره‌ای کلاسیک |
| Computer Program | برنامه کامپیوتری |
| Concatenation | چسباندن |

Conditional شرط

Countably Infinite شمارای نامتناهی

D

Disjunction فصل

E

Empty تهی

Environment محیط

Equivalent هم‌ارز

Execution اجرا

Expressible قابل بیان

Expression عبارت

Expressivity قدرت بیان

F

False غلط

Finite متناهی

Finite State Machine ماشین حالات متناهی

Fixpoint نقطه ثابت

Formal صوری

Formal Method روش صوری

Formal Semantics معنانشناسی صوری

Formalization صوری‌گری

Formula فرمول

تابع Function

I

زبان برنامه نویسی دستوری Imperative Programming Langugae

پیاده‌سازی Implementation

نامتناهی Infinite

J

جاوا Java

L

برچسب Label

کوچکترین نقطه ثابت Least Fixpoint

منطق زمانی خطی Linear Temporal Logic

منطق Logic

حلقه Loop

M

عضو Member

حافظه Memory

ادوات وجهی Modal Connective

منطق موجهات Modal Logic

مدل Model

وارسی مدل Model Checking

Model of Computation مدل محاسبه

N

Natural Language زبان طبیعی

Natural Number عدد طبیعی

Negative Integer عدد صحیح منفی

Non-deterministic غیر قطعی

O

Operator عملگر

P

Pair زوج مرتب

Positive Integers عدد صحیح مثبت

Prefix Trace رد پیشوندی

Prefix Trace Semantics معنانشناسی رد پیشوندی

Processor پردازنده

Property ویژگی

Python پایتون

R

Recursive بازگشتی

Register Machine ماشین رجیستر

S

| | |
|----------------------------|-----------------------|
| Satisfiable | ارضای پذیر |
| Scheduling Algorithm | الگوریتم زمانبندی |
| Semantics | معناشناسی |
| Sequence | دنباله |
| Set | مجموعه |
| Sheffer Operator | ادوات شفر |
| State | وضعیت |
| Statement | عبارت دستوری |
| Statement List | لیست عبارت های دستوری |
| Structure | ساختار |
| Subset | زیر مجموعه |
| Subtraction | منها |
| Syntax | نحو |

T

| | |
|----------------------|--------------|
| Temporal Logic | منطق زمانی |
| True | درست |
| Truth | درستی |
| Turing Machine | ماشین تورینگ |

V

| | |
|--------------------|------------|
| Valid | معتبر |
| Value | مقدار |
| Variable | متغیر |
| Verification | درستی یابی |

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 Amorim and Chris Casinghino, M. Gaboardi, M. Greenberg, C. Hrițcu, 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

Abstract goes here...



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

Thesis Title

Author name

Supervisor: name

Co-Supervisor: name

Advisor: name

A thesis submitted to Graduate Studies Office
in partial fulfillment of the requirements for the degree of
B.Sc./Master of Science/Doctor of Philosophy in
Pure Mathematics/ Applied Mathematics/ Statistics/ Computer Science

yyyy