



دانشکده‌گان علوم  
دانشکده ریاضی، آمار و علوم کامپیوتر

# بهبود روش واریسی مدل با استفاده از نظریه تعبیر مجرد

نگارنده

پویا پرتو

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

پایاننامه برای دریافت درجه کارشناسی ارشد  
در رشته علوم کامپیوتر

تاریخ دفاع

## چکیده

روش وارسی مدل یک روش قابل اعتماد برای بررسی صحت عملکرد برنامه‌های کامپیوتری است. بیان‌های مختلف این روش از منطق موجهات بهره می‌برند که چندان برای برنامه نویسان شناخته شده نیستند. در این رساله سعی شده یک بیان جدید از روش وارسی 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 ارائه مفاهیم اولیه، به سه صورت متفاوت به بیانی جدید از روش وارسی 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: وارسی مدل، نظریه تعبیر مجرد، معناشناسی دلالتی، پیوند گالوا، درستی یابی صوری، تحلیل ایستا، درستی یابی 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 داشته‌باشد که در تست‌ها نیامده‌باشد. 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 مورد بررسی می‌سازیم که همه‌ی حالات اجرای برنامه با آن قابل توصیف 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 بیان شده را بررسی می‌کنیم. مقاله [۴] شروع این روش‌ها بوده که این کار را با استفاده از نوعی مدل کریپکی [۱۷] و نوعی منطق زمانی به نام منطق زمانی خطی [۴] انجام داده که روشی است با دقت و البته هزینه 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 به‌طور تئوری، 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 به‌طور مقدماتی پرداخته شده است. همین‌طور 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
۲	۱.۱.۱ زبان LTL	۲
۳	۲.۱.۱ معنانشناسی LTL	۳
۴	۲.۱ نحو زبان مورد بررسی	۴
۵	۳.۱ معنانشناسی زبان مورد بررسی	۵
۵	۱.۳.۱ برجسبها	۵
۶	۲.۳.۱ رد پیشوندی	۶
۷	۳.۳.۱ تعریف صوری معنانشناسی رد پیشوندی	۷
۱۱	۲ صوری‌گری جدید برای روش واریسی مدل	۱۱
۱۱	۱.۲ ویژگی‌های معنایی برنامه‌ها	۱۱
۱۲	۱.۱.۲ ویژگی‌های معنایی	۱۲
۱۲	۲.۱.۲ عبارات منظم	۱۲
۱۲	۳.۱.۲ زبان عبارات منظم	۱۲
۱۴	۴.۱.۲ معنانشناسی عبارات منظم	۱۴
۱۷	۵.۱.۲ گونه‌های مختلف زبان عبارات منظم	۱۷
۱۸	۲.۲ صورت جدید مسئله‌ی واریسی مدل	۱۸
۱۹	۳.۲ در مورد توقف پذیری	۱۹
۲۵	۳ واریسی مدل منظم	۲۵
۲۵	۱.۳ در مورد عبارات منظم	۲۵
۲۵	۱.۱.۳ هم‌ارزی عبارات منظم	۲۵
۲۶	۲.۱.۳ فرم نرمال فصلی	۲۶
۳۱	۳.۱.۳ سرو دم عبارات منظم	۳۱
۳۶	۲.۳ واریسی مدل منظم	۳۶
۳۶	۱.۲.۳ صورت	۳۶

۳۹	.....	۲.۲.۳ درستی و تمامیت
۴۴		۴ واری مدل ساختارمند
۵۰		۵ نتیجه گیری



## فصل ۱

### مقدمه و مفاهیم اولیه

در این فصل به عنوان مقدمه، روش واریسی مدل به طور مختصر معرفی شده است. در فصل های بعدی، با هدف بهبود این روش، صورت های 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 می شود. محوریت کار ما [۶] است که در آن روش جدید واریسی مدل ارائه شده است. بحث با ارائه ی یک زبان شروع می شود، سپس معناشناسی رد پیشوندی برای این زبان ارائه می شود و فصل تمام می شود. مفاهیم معرفی شده در این فصل دارای ریزه کاری های زیادی هستند و به عقیده ی نگارنده، در [۶] در ارائه ی بعضی از جزئیات سهل انگاری اتفاق افتاده است. سعی کرده ایم که اگر ایرادی در تعاریف موجود در [۶] هست، حین بیان دوباره ی آن 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 درستی یابی سیستم های مختلف استفاده می شود. در این روش معمولاً ابتدا یک ماشین حالات متناهی از روی سیستم مورد بررسی ساخته می شود، سپس بررسی هایی که قرار است روی سیستم اصلی انجام شوند، روی این ماشین (مدل) انجام می شود.

از این روش در بررسی صحت کارکرد برنامه های کامپیوتری استفاده می شود، اما این تنها مورد استفاده ی این روش نیست. و هر سیستمی که قابلیت بیان شدن به صورت صوری را داشته باشد، با این روش قابل بررسی است. مثلاً می توان 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 می دهد) و

می‌توان از روش واریسی مدل برای اطمینان از عدم وجود چنین ویژگی نامطلوبی استفاده کرد. مثال های دیگر استفاده‌ی این روش در علوم کامپیوتر بررسی صحت عملکرد یک پردازنده یا الگوریتم تقسیم وظایف یک سیستم عامل است. این مثال‌ها هیچ کدام یک برنامه‌ی کامپیوتری نیستند (هر چند که ممکن است مجبور باشیم از یک برنامه‌ی کامپیوتری برای پیاده سازی 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 خواص 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 ادات وجهی گوناگون، ساخته می‌شوند. این ادوات غالباً در زبان طبیعی نقش قید را دارند. منطق‌های زمانی دسته‌ای از منطق‌های موجهات هستند که به صوری‌گری ما مفهوم زمان را اضافه می‌کنند، یعنی قیدهایی مانند فعلاً، بعداً، و قبلاً (که مورد آخری کمتر رایج است). منطقی که در اینجا بیان می‌کنیم  $LTL$  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 به‌طور مستقیم بیانگر این قیده‌ها نیستند، هرچند که به کمک ادوات جدید می‌توان ادواتی برای هر یک از این قیود ساخت. این تعاریف از [۱۹] آورده شده‌اند.

ابتدا زبان 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.

## ۱.۱.۱ زبان $LTL$

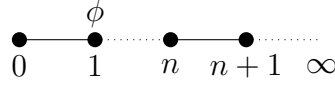
تعریف ۱.۱. هر عضو مجموعه‌ی  $\Phi$  یک فرمول در زبان  $LTL$  است (و  $\Pi$  مجموعه‌ی (شمارای نامتناهی) فرمول‌های اتمی است و  $\pi \in \Pi$ ):

$$\Pi \subset \Phi,$$

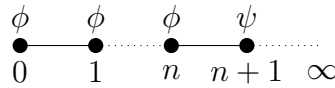
$$\phi \in \Phi \Leftrightarrow \phi ::= \pi \mid \phi \vee \phi \mid \neg \phi \mid \bigcirc \phi \mid \phi \mathcal{U} \phi$$

در این منطق، ما 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 با 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 دو ادوات دیگر را در زبان خود دارد. دلیل وجود ادوات متفاوت، می‌تواند راحت‌تر کردن بیان خواص باشد. همان طور که استفاده نکردن از یا و شرطی در منطق گزاره‌ای می‌تواند به سخت کردن بیان جملات در چارچوب این منطق منجر شود، حذف این ادوات وجهی هم بیان خواص 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 معناسازی 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  $\phi$ ، در لحظه‌ی  $i$  در مدل  $M$  ارزش درست دارد. این مفهوم را، به صورت بازگشتی، به شکل زیر تعریف می‌کنیم:

$$\begin{aligned}
 M, i \models \pi & \text{ iff } \pi \in M(i) \\
 M, i \models \neg\phi & \text{ iff } M, i \not\models \phi \\
 M, i \models \phi \vee \psi & \text{ iff } M, i \models \phi \text{ or } M, i \models \psi \\
 M, i \models \bigcirc\phi & \text{ iff } M, i+1 \models \phi \\
 M, i \models \phi\mathcal{U}\psi & \text{ iff } \exists k \geq i \in \mathbb{N}_0 : \forall i \leq j < k : M, j \models \phi \text{ and } M, k \models \psi
 \end{aligned}$$

یک فرمول را ارضا پذیر می‌گوییم اگر و تنها اگر مدلی وجود داشته باشد که فرمول در آن صادق باشد. اگر یک فرمول در هر مدلی صادق باشد، آن فرمول را معتبر می‌گوییم.

## ۲.۱ نحو زبان مورد بررسی

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

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

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

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

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

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

$$x \doteq A;$$

$$|$$
 ;

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

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

$$| \{S\}$$

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

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

قابل مشاهده است که این زبان، نسبت به کل زبان C، تا حد ممکن ساده سازی 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 ارائه‌ی این زبان صرفاً ارائه‌ی روش جدید است. یعنی می‌توان به این زبان به چشم یک مدل محاسباتی، مانند ماشین تورینگ و ماشین رجیستر، نگاه کرد. روشی که سعی در ارائه‌اش داریم، برای زبان‌های برنامه نویسی دستوری است، مانند پایتون، جاوا و C. بنابراین، انتخاب یک مدل محاسباتی که به 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 یکی اعداد منفی را بسازیم و سپس بعد از آن به سراغ اعداد مثبت می‌رویم که با کمک ۰ و هر عدد منفی‌ای که ساختیم، ساخته می‌شوند. باقی اعداد و حتی باقی عملگرها (یعنی به غیر از اعداد طبیعی) نیز از روی آنچه داریم قابل ساختن است. در مورد عبارت‌های بولی نیز داستان به همین منوال است. یعنی اینجا صرفاً ادات شفر تعریف شده و باقی عملگرهای بولی 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 آن اجرای برنامه را از دستوری بعد از داخلی‌ترین حلقه‌ای که break; داخلش قرار دارد ادامه می‌دهد. در پایان می‌توان ثابت کرد که این زبان هم قدرت با ماشین تورینگ [۹] است.

توجه داریم که هرچه در بالا در مورد معنای دستورات این زبان گفتیم، به هیچ وجه صوری نیست. صرفاً درک شهودی‌ای که از معنای اجرای هریک از دستورات می‌توان داشت را بیان کرده‌ایم. بیان صوری معنای برنامه‌ها را که خلاف درک شهودی‌مان قابل انتقال به کامپیوتر است، در ادامه بیان خواهیم کرد. طبیعتاً، این بیان صوری از روی یک درک شهودی ساخته شده است.

### ۳.۱ معنانشناسی زبان مورد بررسی

معنانشناسی زبانی را که در بخش پیش آورده‌ایم، با کمک مفاهیمی به نام‌های ”برچسب” و ”رد پیشوندی” و عملگری به نام ”چسباندن” تعریف می‌کنیم. نام این معنانشناسی ”معنانشناسی رد پیشوندی” است.

#### ۱.۳.۱ برچسب‌ها

با وجود اینکه در زبان C مفهوم برچسب، که می‌خواهیم معرفی اش کنیم، وجود دارد، اما در زبانی که معرفی کردیم، خبری از برچسب‌ها نبود. با این وجود، برای تعریف صوری معنای برنامه‌ها، به این مفهوم نیاز داریم. در این بخش، به‌طور غیر دقیق معنای برچسب‌ها را آورده‌ایم. همین تعاریف غیر دقیق برای کار ما کافی است. تعاریف صوری دقیق‌تر این موجودات در پیوست [۶] آورده شده‌اند. از آوردن مستقیم این تعاریف در اینجا خودداری کرده‌ایم. البته در مورد معنای صوری برچسب‌ها قابل ذکر است که طبق [۷]، تعریف صوری برچسب‌ها غیر قطعی است. به عبارت دیگر، این تعریف ناکامل است و سیستم‌های صوری متفاوتی را می‌توان متصور شد که در تعریف صوری برچسب‌ها می‌گنجند.

در زبانمان،  $S$  ها بخشی از عبارات موجود در زبان هستند. برچسب ها را برای  $S$  ها تعریف می کنیم. برچسب ها با کمک توابع  $labs, in, brks-of, brk-to, esc, aft, at$  تعریف می شوند. 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 برچسب ها را برگردانند. یکی از آن ها هم با گرفتن  $S$  یک مقدار بولی را برمی گردانند.

$at[S]$ : برچسب شروع  $S$ .

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

$esc[S]$ : یک مقدار بولی را باز می گرداند که بسته به اینکه در  $S$  دستور  $break$  وجود دارد یا خیر، مقدار درست یا غلط را برمی گرداند.

$brk-to[S]$ : برچسبی است که اگر حین اجرای  $S$  دستور  $break$  اجرا شود، برنامه از آن نقطه ادامه پیدا می کند.

$brks-of[S]$ : مجموعه ای از برچسب دستورات  $break$  های داخل  $S$  را برمی گرداند.

$in[S]$ : مجموعه ای از تمام برچسب های درون  $S$  را برمی گرداند.

$labs[S]$ : مجموعه ای از تمام برچسب هایی که با اجرای  $S$  قابل دسترسی هستند را برمی گرداند.

مجموعه ی همه ی برچسب ها را با  $\mathbb{L}$  نشان می دهیم.

### ۲.۳.۱ رد پیشوندی

حال که تعریف برچسب ها را هم داریم، به سراغ تعریف رد پیشوندی می رویم. البته پیش از آن، باید وضعیت ها و محیط ها را تعریف کنیم.

تعریف ۲.۱. (محیط): به ازای مجموعه مقادیر  $\mathbb{V}$  و مجموعه متغیرهای  $\mathbb{X}$  تابع  $\rho: \mathbb{X} \rightarrow \mathbb{V}$  را یک محیط می گوئیم. مجموعه ی همه ی محیط ها را با  $\mathbb{E}\mathbb{V}$  نمایش می دهیم.

تعریف ۳.۱. (وضعیت): به هر زوج مرتب به ترتیب متشکل از یک برچسب  $l$  و یک محیط  $\rho$  یک وضعیت (یا حالت)  $\langle l, \rho \rangle$  می گوئیم. مجموعه ی همه ی وضعیت ها را با  $\mathbb{S}$  نشان می دهیم.

تعریف ۴.۱. (رد پیشوندی): به یک دنباله از وضعیت ها (با امکان تهی بودن) یک رد پیشوندی می گوئیم.

هر رد پیشوندی یک دنباله است که قرار است توصیفی از چگونگی اجرای برنامه باشد. وضعیت ها موقعیت لحظه ای حافظه ای در دسترس برنامه است را توصیف می کنند.  $l$  برچسب قسمتی از برنامه است که در حال اجرا است و  $\rho$  مقدار متغیرها را در آن موقع از اجرای برنامه نشان می دهد. دنباله های ما می توانند متناهی یا نامتناهی باشند. مجموعه ی ردهای پیشوندی متناهی را با

$\mathfrak{S}^+$  و مجموعه‌ی ردهای پیشوندی نامتناهی را با  $\mathfrak{S}^\infty$  نمایش می‌دهیم. مجموعه‌ی همه‌ی ردهای پیشوندی را هم با  $\mathfrak{S}^{+\infty}$  نمایش می‌دهیم. با توجه به آنچه گفتیم، یک عملگر چسباندن  $\bowtie$  را روی ردهای پیشوندی تعریف می‌کنیم.

پیش از ارائه‌ی تعریف، به دو نکته‌ی مهم در مورد نمادگذاری‌های این پایان نامه اشاره می‌کنیم. اولین نکته این است که حین ارائه‌ی تعریف‌ها، مانند تعریف عملگر چسباندن که در ادامه آمده، اگر تعریف را روی یک ساختار یا با در نظر گرفتن پیش فرض‌های مختلف ارائه داده باشیم، ابتدا، هر فرض را با علامت  $\blacktriangleleft$  نشان داده‌ایم. در اثبات‌ها به جای این نماد از  $\blacktriangleright$  استفاده کرده‌ایم. نکته‌ی دوم در مورد نشان دادن ردهای پیشوندی است. اگر  $\pi_1, \pi_2$  رد پیشوندی باشند و  $\sigma$  یک وضعیت باشد،  $\pi_1\sigma$  به یک رد پیشوندی لزوماً متناهی اشاره می‌کند که با وضعیت  $\sigma$  به پایان رسیده است،  $\sigma\pi_1$  به یک رد پیشوندی اشاره می‌کند که با وضعیت  $\sigma$  شروع شده است و  $\pi_1\pi_2$  به یک رد پیشوندی اشاره می‌کند که با  $\pi_1$  شروع شده است و با  $\pi_2$  ادامه پیدا می‌کند ( $\pi_1$  باید متناهی باشد). توجه شود که  $\pi_1\pi_2$  با چسباندن  $\pi_1$  و  $\pi_2$  ( $\pi_1 \bowtie \pi_2$ ) به همدیگر متفاوت است.

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

همینطور،  $\epsilon$  یک رد پیشوندی است که حاوی هیچ وضعیتی نیست. به عبارت دیگر، یک دنباله‌ی تهی است.

### ۳.۳.۱ تعریف صوری معناسازی رد پیشوندی

در این بخش، دو تابع  $A$  و  $B$  را به ترتیب روی عبارات حسابی و بولی زبانمان، یعنی  $A$ ها و  $B$ ها تعریف می‌کنیم، سپس با کمک آنها  $S^*$  را روی مجموعه‌ای از اجتماع معنای  $S$ ها و  $SA$ ها تعریف می‌کنیم. پس در نهایت، هدف ما تعریف  $S^*$  است.

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

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

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

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

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

$$\begin{aligned} \mathcal{B}[A_1 < A_2] \rho &= True && \text{اگر } \mathcal{A}[A_1] \rho \text{ کوچکتر از } \mathcal{A}[A_2] \rho \text{ باشد} \\ \mathcal{B}[A_1 < A_2] \rho &= False && \text{اگر } \mathcal{A}[A_1] \rho \text{ بزرگتر از } \mathcal{A}[A_2] \rho \text{ باشد} \\ \mathcal{B}[B_1 \text{ nand } B_2] \rho &= \neg(\mathcal{B}[B_1] \rho \wedge \mathcal{B}[B_2] \rho) \end{aligned}$$

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

در ادامه، به تعریف  $\mathcal{S}^*$  می‌پردازیم. این کار را با تعریف  $\mathcal{S}^*$  روی هر ساخت  $S$  و  $SI$  انجام می‌دهیم. 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 برچسب  $l$  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 دستور 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 دستور 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 دستور 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 دستور 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  $SI ::= S$  باشد، ردهای پیشوندی متناظر با اجرای این دستور 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 دستور 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 (*Join*) 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، ماجر نسبت به حالات قبل اندکی پیچیده‌تر می‌شود. تابعی به اسم  $\mathcal{F}$  را تعریف خواهیم کرد که دو ورودی دارد. ورودی اول آن یک دستور 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 دستورات داخل حلقه را اجرا می‌کند و دنباله‌هایی جدید را از دنباله‌های قبلی می‌سازد.

معنای یک حلقه را کوچکترین نقطه ثابت این تابع در نظر می‌گیریم. در ادامه، تعریف  $\mathcal{F}$  آمده است. با دیدن تعریف، می‌توان به دلیل این کار پی‌برد. ورودی‌ای که دیگر  $\mathcal{F}$  روی آن اثر نمی‌کند، در دو حالت ممکن است اتفاق افتد. 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

است که اجرای برنامه داخل حلقه به دستور break برخورد کرده است که در آن صورت وضعیتی به ته ردهای پیشوندی اضافه می‌شود که برچسبش خارج از مجموعه برچسب دستورات حلقه است و همین اضافه کردن هر چیزی را به ته ردهای پیشوندی موجود، توسط  $\mathcal{F}$  غیرممکن می‌کند. بنابراین، نقطه ثابت مفهوم مناسبی برای این است که از آن در تعریف صوری معنای حلقه استفاده کنیم. 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 اجرای برنامه مرتبط است و ردهای پیشوندی اضافی و بی‌ربط به معنای برنامه، به آن وارد نمی‌شوند. برای درک بهتر این نکته می‌توان به این نکته توجه کرد که با اضافه کردن وضعیت‌هایی کاملاً بی‌ربط به اجرای 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 چه‌طور می‌توانیم مطمئن باشیم که چنین نقطه ثابتی وجود دارد. در این رابطه، باید گفت که مجموعه‌هایی که از ردهای پیشوندی تشکیل می‌شوند با عملگر زیرمجموعه بودن یک شبکه را تشکیل می‌دهند و بنا به قضیه تارسکی [۲۳] برای چنین موجودی نقطه ثابت وجود دارد. تعاریف موجوداتی که در موردشان صحبت کردیم، به این شکل است:

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

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

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

در اینجا، تعریف معناشناسی 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 خاصیت‌های مورد بررسی به‌جای منطق زمانی از عبارات منظم استفاده می‌شود. با صحبت در مورد ویژگی‌های برنامه‌ها در صوری‌گری‌ای که داریم شروع می‌کنیم، سپس به معرفی عبارات منظم، به عنوان یک وسیله برای بیان ویژگی‌ها، می‌پردازیم و پس از آن، صورت روش واریسی مدل را ارائه می‌کنیم. در آخر این فصل نیز، به بحث در مورد تصمیم‌پذیری این روش می‌پردازیم.

## ۱.۲ ویژگی‌های معنایی برنامه‌ها

تا به اینجا کار، یک زبان آورده‌ایم و برای آن معنا تعریف کرده‌ایم. در این بخش، در مورد ویژگی‌های برنامه‌هایی که در این زبان نوشته می‌شوند، با توجه به معنای صوری‌ای که تعریف کرده‌ایم، صحبت می‌کنیم. برای برنامه‌هایی که در یک زبان برنامه‌نویسی نوشته می‌شوند، می‌توان به اشکال مختلفی ویژگی تعریف کرد. مثلاً ویژگی‌های نحوی، مثل این که 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 پیچیدگی محاسبات بررسی می‌شود. منظور ما در اینجا از تعریف ویژگی، متناسب است با معنانشناسی‌ای که برای برنامه‌هایمان تعریف کرده‌ایم. معنانشناسی‌ای که تعریف کرده‌ایم، در واقع سیر محاسباتی برنامه را توصیف می‌کند و ما می‌خواهیم ویژگی‌ها را با توجه به این موضوع تعریف کنیم. در این صورت، می‌توانیم، صحت عملکرد برنامه‌ها را با توجه به صادق بودن ویژگی‌هایی که در مورد آن‌ها تعریف شده بفهمیم. ابتدا به تعریف ویژگی‌ها می‌پردازیم، سپس به سراغ تعریف یک نوع عبارت منظم می‌رویم که از آن برای بیان ویژگی‌ها استفاده می‌شود.

## ۱.۱.۲ ویژگی‌های معنایی

همان‌طور که در بخش قبلی دیدیم، معنای هر برنامه با یک مجموعه‌ی  $S^*[[S]]$  مشخص می‌شود. وقتی می‌خواهیم ویژگی‌هایی را برای موجوداتی که به کمک مجموعه‌ها تعریف شده‌اند بیان کنیم، این که 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 و این که یک عدد زوج هست یا نه را عضویتش در مجموعه‌ی  $\mathbb{E}$  تعریف کنیم. پس یعنی در مورد اعداد طبیعی، هر ویژگی به شکل زیرمجموعه‌ای از تمام این اعداد در نظر گرفته می‌شود. یعنی هر عضو  $P(\mathbb{N})$  بنا به تعریف ما یک ویژگی از اعداد طبیعی است. در مورد برنامه‌ها نیز قرار است همین رویه را پیش بگیریم. تابع  $S^*$  از نوع  $P(\mathbb{G}^+) \rightarrow \mathbb{P}$  است. 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  $P(\mathbb{G}^+)$  تعریف کنیم، به عبارت دیگر عضوی از  $P(P(\mathbb{G}^+))$ .

## ۲.۱.۲ عبارات منظم

در اینجا، توصیف ویژگی‌ها برای هر برنامه باید یک چارچوب داشته باشد. در صورت قدیمی روش واریسی مدل ما از منطق‌های زمانی برای بیان ویژگی‌ها به صورت صوری استفاده می‌کردیم و این احتیاج به یک زبان برای صوری کردن کامل کار را، که رسیدن به بیان مسئله‌ی واریسی مدل است، به ما نشان می‌دهد. در اینجا ما با داستان دیگری هم رو به رو هستیم و آن این است که از آنجایی که با مجموعه‌ها سر و کار داریم و مجموعه‌ها چندان موجودات ساختنی‌ای نیستند (برخلاف مدل کریپکی)، بهتر است یک موجود ساختنی مثل یک زبان صوری برای بیان آن‌ها داشته باشیم. در این فصل قصد داریم یک نوع عبارت منظم را برای این منظور تعریف کنیم. پیش‌تر به نکته‌ی دیگری در مورد استفاده از عبارات منظم، که متداول‌تر بودن بین جامعه‌ی برنامه نویسان است، صحبت کردیم. ابتدا زبان این عبارت منظم را تعریف می‌کنیم، سپس به سراغ معناشناسی آن می‌رویم.

## ۳.۱.۲ زبان عبارات منظم

فرق عمده‌ای که زبان عبارات منظم ما با عبارات منظم کلاسیک دارد، در کاراکترهاست. کاراکترها در زبان کلاسیک موجوداتی اتمی بودند، اما در اینجا، ساختار دارند. در اینجا، به جای هر کاراکتر یک زوج متشکل از مجموعه‌ی  $L$  شامل برجسب‌ها و عبارت بولی  $B$  تشکیل شده‌اند که این زوج را به شکل  $L : B$  در زبانمان نمایش می‌دهیم. زبان ما به شکل BNF زیر است:

## تعریف ۱.۲.

$$\begin{aligned}
 &L \in P(\mathbb{L}) \\
 &x, y, \dots \in \mathbb{X} \\
 &\underline{x}, \underline{y}, \dots \in \underline{\mathbb{X}} \\
 &B \in \mathbb{B} \\
 &R \in \mathbb{R} \\
 &R ::= \varepsilon \\
 &\quad | L : B \\
 &\quad | R_1 R_2 \quad (or \ R_1 \bullet R_2) \\
 &\quad | R_1 \mid R_2 \quad (or \ R_1 + R_2) \\
 &\quad | R_1^* \\
 &\quad | R_1^+ \\
 &\quad | (R_1)
 \end{aligned}$$

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 معنانشناسی عبارات منظم، خواهیم دید که معنی عملگر یگانی  $^+$  به وسیله عملگر یگانی  $^*$  قابل بیان است. توجه شود که پرانتزها هم جزئی از زبان قرار داده شده‌اند.

همین طور در اینجا، می‌خواهیم از تعدادی عبارات مخفف که در ادامه کارمان را راحت‌تر می‌کنند، صحبت کنیم. منظور از زوج  $B : ?$  همان  $B : \mathbb{L}$  است. عبارت  $B : l$  به جای عبارت  $B : \{l\}$  به کار می‌رود و منظور از عبارت  $B : \neg l$  نیز عبارت  $B : \mathbb{L} \setminus \{l\}$  است.

یک نکته‌ی قابل توجه، با توجه به تعاریف فصل قبل، وجود یک مجموعه به نام  $\underline{\mathbb{X}}$  در کنار  $\mathbb{X}$  که از قبل داشتیم، است. به ازای هر  $x \in \mathbb{X}$  یک  $\underline{x} \in \underline{\mathbb{X}}$  داریم. منظور از  $\underline{x}$  مقدار متغیر  $x$  در ابتدای هر برنامه است. یعنی تابع  $\underline{\rho} : \underline{\mathbb{X}} \rightarrow \mathbb{V}$  که  $\underline{\rho}$  به مجموعه‌ی مقادیر متغیرها است. همان طور که پیش‌تر گفتیم، برای اشاره به یک تابع  $\rho$  از کلمه‌ی ”محیط“ استفاده می‌شود. به همین منوال، برای اشاره به  $\underline{\rho}$  از ”محیط اولیه“ استفاده می‌کنیم. برای اشاره به مجموعه‌ی همه‌ی محیط‌های اولیه هم از نماد  $\underline{\mathbb{V}}$  استفاده می‌کنیم. بقیه‌ی موجودات، از جمله برجسب‌ها و عبارات بولی، را در فصل گذشته تعریف کرده‌ایم.

در ادامه به بیان صوری معنای زبان بیان شده می‌پردازیم.

## 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  $\langle \underline{\rho}, \pi \rangle$  را که  $\pi \in \mathcal{G}^*$  و  $\underline{\rho} \in \mathbb{EV}$  باز می‌گرداند. بنابراین این تابع از نوع  $\mathbb{R} \rightarrow P(\mathbb{EV} \times \mathcal{G}^*)$  است. منظور از  $\mathcal{G}^*$  نیز  $\mathcal{G}^+ \cup \{\epsilon\}$  است.

تعریف استقرایی تابع  $S^r$  به شکل زیر است:

تعریف ۲.۲. تابع  $S^r : \mathbb{R} \rightarrow P(\mathbb{EV} \times \mathcal{G}^*)$  به صورت استقرایی روی ساختار عبارت منظم  $R$  به صورت زیر تعریف می‌شود:

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

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

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

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

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

به‌طوری که، با فرض اینکه دو مجموعه‌ی  $S$  و  $S'$  هر یک معنای یک عبارت منظم باشند:

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

[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.]

$$S^r[R]^0 = S^r[\varepsilon]$$

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

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

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

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

[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.]

$$S^r[(B)] = S^r[B]$$

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

$$\begin{aligned}\mathcal{A}[1]_{\underline{\rho}}, \rho &= 1 \\ \mathcal{A}[x]_{\underline{\rho}}, \rho &= \underline{\rho}(x) \\ \mathcal{A}[x]_{\underline{\rho}}, \rho &= \rho(x) \\ \mathcal{A}[A_1 - A_2]_{\underline{\rho}}, \rho &= \mathcal{A}[A_1]_{\underline{\rho}}, \rho - \mathcal{A}[A_2]_{\underline{\rho}}, \rho \\ \mathcal{B}[A_1 < A_2]_{\underline{\rho}}, \rho &= \mathcal{A}[A_1]_{\underline{\rho}}, \rho < \mathcal{A}[A_2]_{\underline{\rho}}, \rho \\ \mathcal{B}[B_1 \text{ nand } B_2]_{\underline{\rho}}, \rho &= \mathcal{B}[B_1]_{\underline{\rho}}, \rho \uparrow \mathcal{B}[B_2]_{\underline{\rho}}, \rho\end{aligned}$$

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

حال که معناسازی 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 کلایینی 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 کلایینی این بوده است که تعمیمی باشد که این تعاریف نابرابر را در خود جای می‌دهد. در [۱۶] آمده است که برای جبر کلایینی 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 کلایینی شروع کرده است. طبق این تعریف، اگر عبارات منظمی که در اینجا تعریف کرده‌ایم، یک جبر کلایینی می‌بودند، می‌بایستی که برای هر عبارت منظم  $R$  می‌داشتیم  $S^r[\llbracket \varepsilon R \rrbracket] = S^r[\llbracket \varepsilon \rrbracket]$ ، چون  $\varepsilon$  نقش صفر عملگر چسباندن (که عملگر ضرب جبر کلایینی است) را دارد و یک جبر کلایینی برای صفر خاصیت جذب را به عنوان یک اصل دارد. اما در مورد عبارات منظمی که در اینجا تعریف کردیم داریم  $S^r[\llbracket \varepsilon R \rrbracket] = S^r[\llbracket R \rrbracket]$ . بیشتر از این به این بحث نمی‌پردازیم که بحث دامنه‌دار و منحرف کننده‌ایست. یک قضیه را در مورد عبارات منظم ارائه می‌دهیم که پخش پذیری عملگر انتخاب به عملگر چسباندن است و بعد به ادامه‌ی راه اصلیمان می‌پردازیم.

قضیه ۴.۲. برای عبارات منظم  $R, R_1, R_2$  داریم:

$$S^r[\llbracket R \bullet (R_1 + R_2) \rrbracket] = S^r[\llbracket (R \bullet R_1) + (R \bullet R_2) \rrbracket]$$



اثبات.

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

□

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  $\varepsilon$  ها تشکیل شده‌اند.

تعریف ۵.۲. (عبارت منظم تهی -  $\mathbb{R}_\varepsilon$ ):

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

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

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

تعریف ۶.۲. (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  $\varepsilon R_2$  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،  $\varepsilon$  حضور داشته باشد. با این تفصیل می توان دید که دو مجموعه  $\mathbb{R}_\varepsilon$  و  $\mathbb{R}^+$  یک افراز برای مجموعه  $\mathbb{R}$  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 آن ها 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 ۷.۲. (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 ۸.۲. (بستر پیشوندی): اگر  $\Pi \in P(\underline{\mathbb{E}\mathbb{V}} \times \mathbb{G}^+)$ ، آنگاه بستر پیشوندی  $\Pi$  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 ۹.۲. 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  $\Pi$  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 ۱۰.۲. (وارسی مدل): اگر  $\rho \in \mathbb{EV}$ ,  $R \in \mathbb{R}^+$ ,  $P \in \mathbb{P}$  آنگاه:

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

این تعریف بیان می‌کند که اگر برنامه‌ی  $P$  با محیط اولیه‌ی  $\rho$  اجرا شود، این برنامه در صورتی خاصیتی که با عبارت منظم  $R$  بیان شده را دارد که معنای آن زیرمجموعه‌ی بستر پیشوندی 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 معناشناسی برنامه را بتوانیم 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  $\text{prefix}$  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 خاصیت بیان شده با عبارت منظم را دارد. این نکته صرفاً در مورد حضور  $\text{prefix}$  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 خاصیت  $R$  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 ۱۱.۲. (توقف پذیری): برنامه‌ی  $P$  را به همراه اجرای اولیه  $\rho$  توقف پذیر می‌گوییم، اگر و تنها اگر وجود داشته باشد  $\pi \in \mathcal{S}^*[P]$ ، که  $(\rho$  محیط متناظر با محیط اولیه‌ی  $\rho$  است):

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

و اینکه  $\langle aft[P], \rho' \rangle$  در  $\pi$  حضور داشته باشد. این اتفاق را با  $P, \rho \downarrow$  نشان می‌دهیم. همین تعریف را برای لیست دستورات SI یا دستور 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 معنانشناسی‌ای که داریم، تعریف کردن توقف پذیری به معنای وجود رد پیشوندی متناهی در معنای برنامه به هیچ عنوان مناسب نیست، چون معنانشناسی 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 ۱۲.۲. برای برنامه‌ی  $P$  و محیط اولیه‌ی  $\rho$  داریم  $P, \rho \downarrow$ ، اگر و تنها اگر  $\rho$  محیط متناظر با محیط اولیه‌ی  $\rho$  باشد و

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

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  $\rho'$  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:

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

همین طور طبق تعریف برچسب‌ها داریم:

$$at[\varepsilon] = aft[\varepsilon]$$

پس حکم برقرار است.

►  $SI = SI' S$  :

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.

►►  $S = x \doteq A$  ; :

در این حالت، با توجه به تعریف معنای  $S$  که قبل تر ارائه شد، دنباله‌ی

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

در معنای دستور به ازای هر  $\rho$  وجود دارد. پس دنباله‌ای به شکل بالا با محیطی متناظر با  $\rho$  هم در معنای دستور حضور دارد.

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

با توجه به معنای این دستور، دنباله‌ی زیر در معنای این دستور 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 دستور حضور دارد، در حالی که،  $\langle at[S_t], \rho \rangle \pi$  داخل معنای  $S_t$  نیز هست. طبق فرض استقرا می‌دانیم که برچسب آخرین وضعیت  $\pi$  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 دستور 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 دستور حضور دارد:

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

همین‌طور، 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! همان‌طور که پیش‌تر گفتیم، معنای حلقه با استفاده از یک تابع تعریف می‌شود. معنای حلقه کوچکترین نقطه ثابت این تابع است، در حالی که، این تابع وقتی

روی یک مجموعه از ردهای پیشوندی اعمال شود، تاثیرات یک بار اجرای دستورات درون حلقه را روی ردهای پیشوندی درون مجموعه اعمال می‌کند.

طبق تعریف  $\mathcal{F}$ ، مطمئن هستیم که رد پیشوندی‌ای که با محیط  $\rho$  شروع شود، در مجموعه‌ی معنای  $S$  وجود دارد، چون به ازای هر محیط  $\rho$  (نقطه به این خاطر است که با  $\rho$  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  $\langle at[S], \rho \rangle \pi \langle aft[S], \rho' \rangle$  در معنای برنامه قرار خواهد گرفت و می‌توانیم بگوییم اجرای دستور 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  $\rho$  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 دستور 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 دستور مشخص نیست! در [۷] هم در مورد این برچسب‌ها بحث شده است. در آنجا، گفته شده که در مورد بخش هایی از تعریف این توابع، که تعریف مشخصی ارائه نشده است، برداشت آزاد است! ما در اینجا سعی داریم، معقول ترین برداشتی که نسبت به درکمان از این کار می‌توانیم داشته باشیم را بیان کنیم. مهم‌ترین چیزی که در مورد برچسب‌ها در این دستور باید برقرار باشد، این است که اگر این دستور 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 دستورات 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  $S'$  داخلی ترین 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 تعریف

معنای دستورات  $break$ ، رد پیشوندی زیر در معنای این دستور وجود دارد،

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

که نشانه‌ی توقف است.

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

در این صورت، توقف پذیری  $S''$  را از فرض استقرای استقرایی که روی لیست دستورات زده بودیم، داریم، پس  $\{S''\}$  هم توقف پذیر است.

در اینجا، اثبات این طرف قضیه به پایان می‌رسد.

( $\Leftarrow$ ) دوباره باید روی ساختار برنامه‌ها استقرا بزنیم و دوباره چون هر برنامه مساوی با یک لیست از دستورات است، استقرا را ابتدا روی ساختار لیست دستورات و در دل آن روی ساختار خود دستورات انجام می‌دهیم.

در این اثبات به غیر از یک حالت ساختار دستور، که دستور 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 دستور به ازای محیط  $\rho$  در حالت اول متوقف شده است، در واقع فرض کرده‌ایم که در معنای این دستور رد پیشوندی  $\langle at[\![S], \rho \rangle \pi \langle aft[\![S], \rho' \rangle \pi'$  وجود دارد. باید ثابت کنیم، به ازای  $\pi'$  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 معنی غلط می‌داده 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  $\varepsilon$  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  $\rho$  متوقف می‌شود یا خیر. این الگوریتمی است برای مسئله‌ی توقف پذیری، مسئله‌ای که تصمیم ناپذیر است. بنابراین، چنین الگوریتمی نباید وجود داشته باشد که یعنی پیاده سازی‌ای برای 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  $\approx$  تعریف شده روی مجموعه‌ی عبارات منظم یک رابطه‌ی هم‌ارزی است.

اثبات. برای هر عبارت منظم  $R$  داریم:

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

پس این رابطه انعکاسی است.  
اگر  $R_1, R_2 \in \mathbb{R}$  آنگاه داریم:

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

پس این رابطه تقارنی هم هست.  
اگر  $R_1, R_2, R_3 \in \mathbb{R}$  داریم:

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

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

□

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  $\approx$  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  $R \approx R'$  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 هم‌ارز است.

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

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

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

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

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

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

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

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. روی ساختار  $R$  استقرا می‌زنیم.

$$\blacktriangleright R = \varepsilon :$$

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

که  $\varepsilon$  یک فرم نرمال فصلی است.

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

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

که  $L : B$  هم یک فرم نرمال فصلی است.

$$\blacktriangleright R = R_1 R_2 :$$

فرض استقرا این است که  $\text{dnf}(R_1) = R_1^1 + R_1^2 + \dots + R_1^{n_1}$  و  $\text{dnf}(R_2) = R_2^1 + R_2^2 + \dots + R_2^{n_2}$  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  $\mathbb{R}^+$  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  $\mathbb{R}^+$  است.

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

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.

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

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

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

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  $\mathbb{R}^+$  است. علاوه بر این، از فرض استقرا داریم:

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

پس با توجه به تعریف  $\text{dnf}$  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  $R' \in \mathbb{R}^+$   $(\text{dnf}(R_1)) = R' \in \mathbb{R}^+$  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 می دهیم 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  $\text{dnf}(R_1) = R_1^1 + R_1^2 + \dots + R_1^m$  و  $\text{dnf}(R_2) = R_2^1 + R_2^2 + \dots + R_2^n$ .

$$\blacktriangleright R = \varepsilon :$$

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

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

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

$$\blacktriangleright R = R_1 R_2 :$$

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

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

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

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:

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

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

: ( $\subseteq$ )

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

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

طبق فرض استقرا، داریم:

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

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

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

►  $R = R_1 + R_2$  :

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

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

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

(به کمک فرض استقرا)

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

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

►  $R = R_1^*$  :

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

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

(طبق نتیجه‌ای که از قضیه‌ی قبل گرفتیم)

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

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

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

►  $R = R_1^+$  :

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

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

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

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

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

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

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

(طبق فرض استقرا):

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

$$\mathcal{S}^r[(R_1)]$$

□

### ۳.۱.۳ سر و دم عبارات منظم

در این بخش، تابعی را روی عبارات منظم تعریف می کنیم که یک عبارت منظم را می گیرد و یک زوج از عبارات منظم را تحویل می دهد، سپس به بیان یک قضیه در مورد این تابع می پردازیم. این تابع را با  $\text{fstnxt}$  نشان می دهیم. 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 می شود، جدا شود. تابع روی عبارات منظم تهی و عبارات منظمی که عملگر + را دارند تعریف نمی شود.

تعریف ۸.۳. (تابع سر و دم): تابع سر و دم را از نوع  $\mathbb{R} \times \mathbb{R} \rightarrow (\mathbb{R}^+ \cap \mathbb{R}^+)^{\dagger}$  به شکل زیر تعریف می کنیم:

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

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

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

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

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

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

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

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

از این تعریف در صورتی از واریسی 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 روی ساختار گرامر بالا استقرا بزنیم:

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

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 می‌شود.

$$\blacktriangleright R = R_1 R_2 :$$

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

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

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

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

مثل مورد قبل اثبات می‌شود.

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



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

$$\blacktriangleright R = \varepsilon :$$

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

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

چون به ازای هیچ عبارت منظم  $R_1$  ای  $R_1^*$  داخل  $\mathbb{R}^+$  نمی‌افتد، 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.

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

شبهه 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  $\text{fstnxt}(R) = \langle L : B, R' \rangle$  آنگاه  $R' \in \mathbb{R}^{\dagger}$  و  $R \approx L : B \bullet R'$ .

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

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

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  $\varepsilon$  عضو  $\mathbb{R}^+$  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  $\text{fstnxt}(R) = \langle L : B, R'_2 \rangle$  پس  $L : B \bullet R'_2 \approx R_2$  و  $R'_2 \in \mathbb{R}^+$  آنگاه  $\text{fstnxt}(R_2) = \langle L : B, R'_2 \rangle$  که همان طور که گفتیم طبق فرض استقرا  $R'_2 \in \mathbb{R}^+$  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 این که  $R'_1 \in \mathbb{R}_\varepsilon$  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  $R \in (\mathbb{R}^+ \cap \mathbb{R}^+)$  پس چون  $R_2$  زیر رشته‌ی  $R$  است، پس  $R_2 \in \mathbb{R}^+$  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  $R \in \mathbb{R}^+ \cap \mathbb{R}^+$ ، پس زیر رشته‌های آن نیز عملگر  $+$  را نخواهند داشت، پس  $R \in \mathbb{R}^+$ . 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  $\varepsilon$  باشد، حالاتی که بررسی کرده‌ایم اتفاق می‌افتند. اگر هر دو عبارت منظم برابر با  $\varepsilon$  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  $\langle L : B, R'_1 \rangle = \text{fstnxt}(R_1)$ ، 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  $\langle L : B, R'_1 \bullet R_2 \rangle = \text{fstnxt}(R)$ . 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 می‌شود.

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

در این صورت  $\langle L : B, R_2 \rangle = \text{fstnxt}(R)$ . مثل حالت‌های قبل ثابت می‌شود 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  $\langle L : B, R'_1 \rangle = \text{fstnxt}(R_1)$ ، 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،  $\langle L : B, R_1^* \rangle = \text{fstnxt}(R)$  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).

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

(هم‌ارزی وسطی به خاطر این است که  $R'_1$  عضو  $\mathbb{R}^\dagger$  است. اگر یکی از دو هم‌ارزی دیگر هم برقرار نباشند کلاً عملگر \* خوش تعریف نخواهد بود، پس این دو هم‌ارزی 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 کرده‌ایم، در این حالت داریم  $\text{fstnxt}(R) = \langle L : B, R'_1 \bullet R_1^* \rangle$ . به همان دلیل مورد قبلی، می‌دانیم که  $R_1^*$  عضو  $\mathbb{R}^+$  است و طبق فرض استقرا داریم  $R'_1 \in \mathbb{R}^+$ . بنابراین داریم  $R'_1 \bullet R_1^* \in \mathbb{R}^+$ . 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 می‌شود.  $\square$

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

## ۲.۳ واریسی مدل منظم

همان‌طور که پیشتر گفتیم، می‌خواهیم 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 آن‌ها احتیاج داریم، پرداخته‌ایم. در این بخش ابتدا این صورت جدید را بیان می‌کنیم و سپس اثبات می‌کنیم که صورت جدید با صورت قبلی معادل است. همان‌طور که پیشتر هم اشاره شد، تفاوت این 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 ورودی‌اش، یک زوج متشکل از یک محیط اولیه و یک عبارت منظم را در کنار یک برنامه می‌گیرد و در خروجی، همگی ردهای پیشوندی موجود در معنای برنامه را که با عبارت منظم سازگار هستند، داخل یک مجموعه برمی‌گرداند. اما در این بین، مفهوم سازگاری یک رد پیشوندی با یک عبارت منظم چگونه مشخص می‌شود؟ این نکته‌ای است که تا به حال در مورد آن بحث نکرده‌ایم و حالا می‌خواهیم، تعریف تابع  $M^t$  را با این هدف به بحث وارد کنیم. البته، این تابع یک ویژگی بیشتر هم دارد. ویژگی دیگر است که اگر عبارت منظم با رد پیشوندی سازگار نباشد، تابع به ما می‌گوید که کجای عبارت منظم ناسازگاری وجود داشته است. همین‌طور 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  $(\mathbb{B} \times \mathbb{R}^{\dagger}) \rightarrow \mathfrak{G}^{+\infty} \rightarrow (\mathbb{E}\mathbb{V} \times \mathbb{R}^{\dagger})$  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  $\mathbb{R}_{\varepsilon}$  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  $(\mathbb{E}\mathbb{V} \times \mathbb{R}^{\dagger}) \rightarrow P(\mathfrak{G}^{+\infty}) \rightarrow P(\mathfrak{G}^{+\infty} \times \mathbb{R}^{\dagger})$  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  $(\mathbb{EV} \times P(\mathfrak{S}^{+\infty})) \rightarrow P(\mathfrak{S}^{+\infty}) \rightarrow (\mathbb{EV} \times P(\mathfrak{S}^{+\infty}))$  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 خاصیت  $R \in \mathbb{R}$  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:

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

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

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

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

اثبات. اگر زوج  $\langle \underline{\rho}, \pi \rangle$  عضو  $\mathcal{M}\langle \underline{\rho}, R \rangle \mathcal{S}^* \llbracket P \rrbracket$  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:

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

که طبق تعریف  $\mathcal{M}^t$  یعنی:

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

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

□

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 مثلاً 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 اولیه 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 خاصیت  $R$  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  $M^{\uparrow}$  می‌تواند به ما بگوید. محل ناسازگاری عبارت منظمی است که زوج عضو ماکسیمال دسته‌ی مورد نظر در خروجی یکی از اعمال های  $M^{\uparrow}$  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  $\rho$  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:

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

ثابت کنیم:

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

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

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

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

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

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

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

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:

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

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

$$R \bullet (? : T)^* \approx (R_1 + R_2 + \dots + R_m) \bullet (? : T)^*$$

$$\approx R_1 \bullet (? : T)^* + R_2 \bullet (? : T)^* + \dots + R_n \bullet (? : T)^*$$

چون  $R_k \approx \varepsilon$ ، داریم:

$$R_1 \bullet (? : T)^* + R_2 \bullet (? : T)^* + \dots + R_k \bullet (? : T)^* + \dots + R_n \bullet (? : T)^*$$

$$\approx R_1 \bullet (? : T)^* + R_2 \bullet (? : T)^* + \dots + \varepsilon \bullet (? : T)^* + \dots + R_n \bullet (? : T)^*$$



و از طرف دیگر داریم:

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

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

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

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  $\text{fstnxt}(R_k) = \langle L_k^1 : B_k^1, R_k^1 \rangle$ . همین‌طور 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  $R_k^1 \approx \varepsilon$ )، فرض می‌کنیم کاری که انجام دادیم را می‌توانیم روی دم خروجی عبارت منظم  $R_k$  (یعنی  $R_k^1$ ) تکرار کنیم:

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

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

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

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.

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

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:

$$\forall i : 1 \leq i \leq n \rightarrow \langle \underline{\rho}, \pi \rangle \notin \mathcal{M}^t(\underline{\rho}, R_i) \mathcal{S}^*[\mathbb{P}]$$

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

در این صورت، خواهیم داشت:

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

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

## فصل ۴

### وارسی مدل ساختارمند

در این فصل، به ادامه‌ی ساختارمندتر کردن کار می‌پردازیم. در فصل گذشته، ساختار عبارات منظم را به تعریف وارسی مدل اضافه کردیم و حالا می‌خواهیم، ساختار زبانمان را به کار اضافه کنیم. این آخرین تلاش [۶] برای گسترش کار بوده است. یعنی وارسی مدل به شکل جدید تعریف شده است و معادل بودن آن با صورت قبلی وارسی مدل ثابت شده است و پس از آن کار پایان می‌پذیرد. تعریف صورت جدید، چونکه 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 و صرفاً ممکن است، برای بعضی موارد آن یک گزارش کلی از اینکه هر برقراری چگونه ثابت شده است، به سبک [۶] بعد از بیان هر مورد از تعریف، آورده‌باشیم.

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

در ادامه، ممکن است بعضی جاها 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  $\mathbb{P} \cup \mathbb{S} \cup \mathbb{S}$  انجام شده است. یعنی در واقع روی همه‌ی اعضای تعریف هر یک از این سه بخش زبان تعریف کردن صورت گرفته، تقریباً کاری شبیه به اثبات لمی که در بخش ۳.۳ داشتیم. در ادامه موردهای 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  $P$ .

$$\triangleleft P = SI :$$

$$\hat{M}(\underline{\rho}, R)S^*[P] = \bigcup_{i=1}^n \{ \langle \underline{\rho}, \pi \rangle \mid \exists R' \in \mathbb{R}, \langle \pi, R' \rangle \hat{M}^\dagger(\underline{\rho}, R_i)S^*[SI] \}$$

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

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

در ادامه با توجه به تعریف قبل به بیان تعریف  $\hat{M}^\dagger$  پرداخته شده. این تنها بخش تابع  $\hat{M}$  است که معرفی نشده و با مشخص شدن آن معنای  $\hat{M}$  به ازای برنامه‌های مختلف مشخص می‌شود. این نکته را در نظر داشته باشیم که  $\hat{M}^\dagger$  قرار است در عمل روی مجموعه‌ی برنامه‌ها تعریف شود و مثلاً اینکه به ازای  $\Pi \in \mathcal{G}^{+\infty}$  دلخواه که مساوی معنی یک برنامه نیست، این تابع با یک محیط اولیه و یک عبارت منظم چه خروجی‌ای دارد برای ما اهمیتی ندارد و البته به این شکلی هم که تابع تعریف شده حتی به ازای چنین ورودی‌ای خروجی ندارد. در واقع حتی به ازای  $S := x \doteq A$  هم که یک عضو از زبان است، به این معنا که  $S$  صرفاً یک بخش از یک برنامه است و خروجی‌ای نخواهد داشت و داشتن خروجی به ازای این متغیر را از  $\hat{M}^\dagger$  انتظار داریم. مشابه  $M^\dagger$  خروجی  $\hat{M}^\dagger$  هم یک زوج مرتب شامل  $\pi$  ای که  $R$  را ارضا کرده و یک عبارت منظم بدون  $+$  که بخشی از  $R$  را نشان می‌دهد که با  $\pi$  تطابق داده نشده (چون احتمالاً  $\pi$  کوتاه‌تر بوده یا ممکن است اصلاً این عبارت منظم تهی باشد).

$$\blacktriangleleft \hat{M}^\dagger(\underline{\rho}, \varepsilon)[[S]] = \{ \langle \pi, \varepsilon \rangle \mid \pi \in \mathcal{S}^* \} S$$

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

$$\blacktriangleleft \hat{M}^\dagger(\underline{\rho}, R)[[SI]] = \hat{M}^\dagger(\underline{\rho}, R)[[SI']] \cup$$

$$\{ \langle \pi \langle \text{at}[[S]], \rho \rangle \pi', R'' \rangle \mid \langle \pi \langle \text{at}[[S]], \rho \rangle, R' \rangle \in \hat{M}^\dagger(\underline{\rho}, R)[[SI']] \wedge$$

$$\langle \langle \text{at}[[S]], \rho \rangle \pi', R'' \rangle \in \hat{M}^\dagger(\underline{\rho}, R')[[S]] \}$$

از اینجا به بعد کار با تعاریف طولی‌تری از آنچه تا حالا داشتیم، روبرو هستیم، مانند همین تعریف بالا. اما به هرحال مفهوم چندان پیچیده‌ای پشت این تعاریف نیست. تعریف بالا به طور خلاصه می‌گوید همه‌ی ردهای پیشوندی‌ای که در  $\hat{M}^\dagger(\underline{\rho}, R)[[SI']]$  هستند به همراه همه‌ی زوج‌هایی که مولفه اول آن‌ها یک رد پیشوندی است که تکه‌ی اول آن داخل  $\hat{M}^\dagger(\underline{\rho}, R)[[SI']]$  افتاده و ادامه‌ی آن داخل  $\hat{M}^\dagger(\underline{\rho}, R')[[S]]$  است، در حالیکه  $R'$  هم ادامه‌ی عبارت منظم  $R$  است، که به تطبیق با رد پیشوندی داخل  $\hat{M}^\dagger(\underline{\rho}, R)[[SI']]$  که زوج خودش است نرسیده و مولفه‌ی دوم این زوج مرتب نیز ادامه‌ی عبارت منظم  $R$  است که به تطبیق با رد پیشوندی‌ای که در مولفه‌ی اول بود، نرسیده.

برای  $R \in \mathbb{R}^\dagger \cap \mathbb{R}^+$  و  $SI = \varepsilon$  داریم:

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

یعنی همه‌ی ردهای پیشوندی تک عضوی که محیطشان اولین لبترال موجود در عبارت منظم را ارضا می‌کند. در واقع هر مخیطی که این لبترال را ارضا کند، برچسب این مجموعه دستور را در این مجموعه می‌آورد (به همراه باقی عبارت منظم).  
برای  $R \in \mathbb{R}^\dagger \cap \mathbb{R}^+$  و  $S = x \doteq A$ ; داریم:

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

با نگاه به این تعریف می‌توان بهتر متوجه شد که اینکه می‌گفتیم قرار است ساختار زبان وارد صورت فصل پیش شود یعنی چه. به جای اینکه مانند فصل گذشته نسبت به ردهای پیشوندی متفاوت خروجی تابع تعیین شود، نسبت به دستوری که معنایش ردهای پیشوندی هستند، خروجی تعیین می‌شود.

در اینجا هم با اینکه تعریف متکلف است اما معنای ساده‌ای دارد. این تابع دستور را به همراه زوج محیط اولیه و عبارت منظم می‌گیرد، ابتدا همان چیزهایی را که تابع در حالت قبلی برمی‌گرداند را برمی‌گرداند و سپس نسبت به اینکه پس از تغییر در محیط‌ها در اثر اجرای دستور مقدار دهی ادامه‌ی عبارت منظم سازگار هست یا نه زوج‌های متشکل از رد پیشوندی و عبارت منظم را به خروجی اضافه می‌کند. نکته‌ای که به جزئیات این تعریف اضافه کرده این است که برای اینکه ادامه‌ی عبارات منظم خارج شده بعد از بررسی اولین لبترال عبارت منظم، یعنی  $R'$ ، آیا تهی است یا نه و این دو حالت متفاوت را در بر خواهد گرفت که در تعریف لحاظ شده‌اند.  
از این ۴ حالت تنها اثبات حالت آخر در [۶] آورده شده و خب اثبات دیگر حالات را هم می‌تواند در همین اثبات که مفصل‌تر و کلی‌تر است، دید. اثبات سر راست است، از جایگذاری تساوی‌های واضح استفاده شده و جزئیات و توضیحات کافی دارد.

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

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

با توجه به موارد قبلی می‌شود مفهوم این دو مورد از تعریف را هم فهمید و پیچیدگی بیشتری ندارد. در باره‌ی مورد اول داستان به این شکل است که مانند تعریف‌های قبلی ابتدا تطابق لیترال اول عبارت منظم با ردهای پیشوندی تک عضوی با برجسب شروع دستور و محیط دلخواه بررسی می‌شود، سپس به این‌ها ردهای پیشوندی‌ای که در درون دستور  $S_t$  هستند و در موقعیت ابتدایشان عبارت بولی معنای صحیح دارد و البته با اولین لیترال باقی عبارت منظم هستند هم در زوج‌هایی

که از یک دانه از ردهای پیشوندی و یک عبارت منظم که بخش تطبیق داده نشده‌ی  $R$  با رد پیشوندی تشکیل شده‌اند، به مجموعه‌ی خروجی اضافه می‌شوند. دو مجموعه‌ی دیگری که در اینجا با خروجی‌ای که تاحالا توصیف کرده‌ایم اجتماع گرفته شده‌اند هم مربوط به حالتی است که در ابتدای رد پیشوندی عبارت بولی معنای غلط دارد و در این صورت چون برخلاف حالت قبل که عبارت منظم تطبیق داده نشده با خود تابع  $\hat{\mathcal{M}}^\dagger$  مشخص می‌شد، در این حالت به ازای اینکه این ادامه‌ی عبارت منظم تهی است یا خیر دو حالت را برای تعریف آن داریم که در واقع عضو دوم زوج مرتب‌های موجود در خروجی را مشخص می‌کنند.

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

$$\blacktriangleleft \hat{\mathcal{M}}^\dagger(\underline{\rho}, R)[S] =$$

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

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

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

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

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

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

با توجه به موارد قبلی این که 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:

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

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

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

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

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

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

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

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

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



$$\begin{aligned}
& \langle \underline{\rho}, \langle at[S], \rho \rangle \rangle \in \mathcal{S}^r[L' : B'] \wedge \langle L'' : B'', R' \rangle = \text{fstnxt}(R''') \wedge \\
& \quad \langle \underline{\rho}, \langle aft[S], \rho \rangle \rangle \in \mathcal{S}^r[L'' : B''] \} \\
& \cup \{ \langle \pi_2 \langle at[S], \rho \rangle \langle at[S_b], \rho \rangle \pi_3, \varepsilon \rangle | \langle \pi_2 \langle at[S], \rho \rangle, \varepsilon \rangle \in X \wedge \\
& \quad \mathcal{B}[B]\rho = T \wedge \langle at[S_b], \rho \rangle \pi_3 \in \mathcal{S}^*[S_b] \} \\
& \cup \{ \langle \pi_2 \langle at[S], \rho \rangle \langle at[S_b], \rho \rangle \pi_3, \varepsilon \rangle | \langle \pi_2 \langle at[S], \rho \rangle, R'' \rangle \in X \wedge \\
& \quad \mathcal{B}[B]\rho = T \wedge R'' \notin \mathbb{R}_\varepsilon \wedge \langle L : B, \varepsilon \rangle = \text{fstnxt}(R'') \wedge \\
& \quad \langle \underline{\rho}, \langle at[S], \rho \rangle \rangle \in \mathcal{S}^r[L : B] \wedge \langle at[S_b], \rho \rangle \pi_3 \in \mathcal{S}^*[S_b] \} \\
& \cup \{ \langle \pi_2 \langle at[S], \rho \rangle \langle at[S_b], \rho \rangle \pi_3, R' \rangle | \langle \pi_2 \langle at[S], \rho \rangle, R'' \rangle \in X \wedge \\
& \quad \mathcal{B}[B]\rho = T \wedge R'' \notin \mathbb{R}_\varepsilon \wedge \langle L : B, R'' \rangle = \text{fstnxt}(R'') \wedge \\
& \quad \langle \underline{\rho}, \langle at[S], \rho \rangle \rangle \in \mathcal{S}^r[L : B] \wedge R'' \notin \mathbb{R}_\varepsilon \wedge \\
& \quad \langle L' : B', R''' \rangle = \text{fstnxt}(R''') \wedge \langle \underline{\rho}, \langle at[S_b], \rho \rangle \rangle \in \mathcal{S}^r[L' : B'] \wedge \\
& \quad \langle \langle at[S_b], \rho \rangle \pi_3, R' \rangle \in \hat{\mathcal{M}}^\dagger \langle \underline{\rho}, R'' \rangle [S_b] \} \\
& \text{and } \langle L' : B', R' \rangle = \text{fstnxt}(R)
\end{aligned}$$

برای دستور تفاوتی که با سایر دستورات وجود دارد، حضور یک تابع در تعریف آن است و 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 وقتی که می‌خواهیم ساختار زبان را به صورت واریسی مدل اضافه انتظار می‌داشتیم که سر و کله‌ی عملگر نقطه ثابت هم پیدا شود، همان طور که تعریف بقیه‌ی دستورات زبان مطابق تعریف معانی‌شان که در ابتدای کار آورده‌ایم، انجام شده. برای عبارت منظم  $R \in \mathbb{R}^\dagger \cup \mathbb{R}^+$  و  $S = ;$  داریم:

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

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

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

همین طور صادق بودن یک خاصیت  $R \in \mathbb{R}$  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:

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

در اینجا تعریف واریسی مدل ساختارمند به پایان می‌رسد.

## فصل ۵

### نتیجه گیری

## واژه‌نامه فارسی به انگلیسی

## واژه‌نامه انگلیسی به فارسی

# 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 board 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 logic<sup>1</sup>. *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.
- [25] P. Wolper. Temporal logic can be more expressive. *Inf. Control.*, 56(1/2):72–99, January/February 1983.

## **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