پردیس علوم
دانشکده ریاضی، آمار و علوم کامپیوتر

بهبود روش وارسی 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 شرح داده شود که در آن به کمک نظریه تعبیر مجرد به جای منطق موجهات از عبارات منظم استفاده شده.

در این نوشته ابتدا به بیان مفاهیم اولیه برای بیان جدید روش وارسی مدل پرداخته‌ایم سپس این روش را به سه صورت متفاوت بیان کرده‌ایم. صورت اول ساختار خاصی ندارد و صرفا روش در ادبیاتی نو بیان شده، صورت دوم ساختار عبارات منظم را به صورتبندی‌اش اضافه کرده و صورت سوم ساختار برنامه‌ها را به صورتبندی اضافه کرده تا روشْ به پیاده سازی نزدیک شود و معادل بودن این سه صورت نشان داده شده.

کلمات کلیدی: وارسی مدل، نظریه تعبیر مجرد، معناشناسی دلالتی، پیوند گالوا، درستی یابی صوری، تحلیل ایستا، درستی یابی برنامه‌های کامپیوتری

تقدیم به

تقدیم به

سپاسگزاری

سپاسگزاری

پیشگفتار

با توجه به پیشرفت روز افزون علوم کامپیوتر و ورود کاربردهای آن به زندگی روزمره، پیشرفت در روش‌های ساخت و نگه‌داری برنامه‌ها نیازی آشکار به نظر می‌رسد. یکی از مسائل مهم در این زمینه بررسی صحت کارکرد برنامه‌های نوشته‌شده است. عدم صحت کارکرد برنامه‌های نوشته‌شده بسته به حساسیت کاری که یک برنامه انجام می‌دهد می‌تواند تبعات مختلفی داشته‌باشد. پرتاب ناموفق آریان ۵[۱۷] ، از مدار خارج شدن مدارگرد مریخ [۲] و تصادف هلیکوپتر چینوک [۱] چند نمونه از تبعات بزرگ این قضیه در گذشته بوده اند، هرچند که به‌سادگی می‌توان فجایع دیگری را مشابه این اتفاقات در زندگی روزمره‌ی انسان‌ها متصور شد. برای کشف صحت کارکرد برنامه‌های کامپیوتری روش های متفاوتی ابداع شده‌اند که در ادامه به طور مختصر از آن‌ها یاد می‌کنیم اما پیش از آن به یک خاصیت مشترک همه ی این روش‌ها می‌پردازیم که ناکامل بودن است. 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. و البته قضیه رایس [۲۱] به ما این تضمین را داده که روش کاملی اصلا وجود ندارد. قضیه رایس به طور غیر رسمی بیان می‌کند که مسئله ی بررسی هر خاصیت غیر بدیهی، برای همه ی برنامه‌ها، تصمیم ناپذیر است. این دلیلی بر این شده که روش‌های مختلفی برای این کار درست شوند که هر کدام می‌توانند حالت‌های خاصی از مسأله را حل بکنند.

یک دسته‌بندی برای این روش‌ها تقسیم آن‌ها به دو دسته‌ی پویا و ایستا است. روش‌های پویا روش‌هایی هستند که در آن‌ها تست برنامه با اجرای برنامه همراه است و روش‌های ایستا بدون اجرای برنامه‌ها انجام می‌شوند. روش‌های پویا معمولاً با اجرای حالات محدودی از برنامه، تصمیم می‌گیرند که برنامه‌ای که نوشته‌ای ایم، انتظاراتمان را برآورده می‌کند یا خیر. اگر این روش بتواند تشخیص دهد برنامه‌ای درست کار نمی‌کند می‌توانیم با اطمینان نتیجه بگیریم که برنامه غلط نوشته‌شده، اما اگر برنامه‌ای از تست‌های ساخته‌شده با این روش‌ها با موفقیت عبور کند، نمی‌توان اطمینان حاصل کرد که برنامه درست کار می‌کند، زیرا ممکن است حالتی مشکل‌زا از اجرای برنامه وجود داشته باشد که در تست ها نیامده باشد. در کتاب [۱۹] به توضیح این روش‌ها پرداخته شده. این دسته از روش‌ها از موضوع اصلی کار ما دور هستند. روش‌های ایستا معمولاً روش‌هایی هستند که از

نظریه‌های مختلف در منطق ریاضی به عنوان ابزار بهره می‌برند تا بدون اجرای خود برنامه‌ها در مورد صحت اجرای آن‌ها نتیجه‌گیری کنند. به همین دلیل به بخشی مهم و بزرگی از این دستورات که از منطق استفاده می‌کنند روش‌های صوری هم گفته‌می‌شود. از معروف ترین این روش‌ها وارسی مدل، روش‌های استنتاجی و استفاده از نظریه تعبیر مجرد است. در روش وارسی مدل، یک مدل صوری متناهی از برنامه‌ی موردبررسی می‌سازیم که همه‌ی حالات اجرای برنامه با آن قابل‌توصیف است، سپس با استفاده از یک زبان صوری که بتواند در مورد مدل هایمان صحبت‌کند، ویژگی‌های مورد بررسیمان را بیان‌می‌کنیم و در نهایت صحت ویژگی‌های بیان‌شده را بررسی می کنیم. مقاله [۴] شروع این روش‌ها بوده که این کار را با استفاده از نوعی مدل کریپکی [۱۶] و نوعی منطق زمانی به نام منطق زمانی خطی [۴] انجام داده که روشی است با دقت و البته هزینه‌ی محاسباتی بسیار بالا. [۱۲] یک منبع بسیار مقدماتی در این زمینه و کتاب[۵] یک مرجع سنتی در این زمینه است. در روش‌های استنتاجی که شاید بتوان یکی از ابتدایی ترین آن‌ها را استفاده از منطق هور[۱۱] دانست، درستی کارکرد برنامه‌هایمان را با ارائه‌ی یک درخت اثبات در یک دستگاه استنتاجی که متناسب با زبان برنامه‌هایمان ساخته‌شده، 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 چون این یک مسئله‌ی NP-Hard است. در[۱۲] به منطق هور به طور مقدماتی پرداخته شده. همین‌طور کتاب[۲۰] نیز به پیاده‌سازی منطق هور در زبان coq پرداخته. coq نیز یک اثبات‌یار است که بر اساس یک نظریه نوع وابسته کار می‌کند. برای اطلاعات بیشتر در مورد چگونگی طرز کار این اثبات‌یار و تئوری بنیادین آن می شود به کتاب[۳] مراجعه‌کرد. تئوری مورد شرح در[۱۰] نیز می‌تواند در این مسیر به کار گرفته شود. نظریه تعبیر مجرد[۸] نیز یک نظریه ریاضیاتی است که در این بحث می‌توان گفت، به‌نوعی سعی می‌کند از روی معناشناسی یک برنامه‌ی کامپیوتری[۲۳] ، 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 هم ساده‌تر از معناشناسی اصلی کار کند تا بتوانیم خواص آن را راحت‌تر بررسی کنیم. در این صورت هر نتیجه‌ای که در مورد خواص جدید بگیریم را می‌توانیم در مورد خود برنامه هم بیان کنیم اما می‌دانیم که در این صورت هم به همه‌ی حقایق دست پیدا نکرده ایم. در مورد این نظریه نیز به‌تازگی کتاب[۷] منتشر شده که حاصل نزدیک به ۵ دهه کار مبدع این نظریه، پاتریک کوزو، است. همین‌طور[۱۴] نیز در مورد پیاده‌سازی این نظریه بحث کرده.

فهرست مطالب

فصل ۱

برخی مفاهیم اولیه

در این فصل سعی شده خواننده با مفاهیم مقدماتی کار آشنا شود. بعضی از مفاهیم ممکن است مستقیما در ادامه استفاده شده باشند و بعضی دیگر ممکن است مستقیما در ادامه وارد بحث نشده باشند اما آشنایی با آن‌ها برای درک بهتر واجب است.

۱.۱ نظریه تعبیر مجرد

نظریه ترتیب مجرد

تعریف ۱.۱. (ترتیب جزئی): اگر $\mathbf{P}$ یک مجموعه باشد و $\leq$ یک رابطه روی این مجموعه باشد به‌طوریکه:

$$1.\forall \mathbf{a} \in \mathbf{P} : \mathbf{a} \leq \mathbf{a}$$

$$2.\forall \mathbf{a}, \mathbf{b} \in \mathbf{P} : (\mathbf{a} \leq \mathbf{b} \wedge \mathbf{b} \leq \mathbf{a}) \rightarrow \mathbf{a} = \mathbf{b}$$

$$3.\forall \mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbf{P} : a \leq b \wedge b \leq c \rightarrow a \leq c$$

آنگاه به زوج $(\mathbf{P}, \leq)$ یک ترتیب جزئی می‌گوییم.

در ادامه خواهیم دید که معنای برنامه‌های کامپیوتری که به یک زبان برنامه نویسی نوشته می‌شوند را می‌شود به شکل یک ترتیب جزئی دید. ترتیب‌های جزئي به عنوان یک نظریه ریاضی مطالعه شده‌اند و اگر معناشناسی یک برنامه را بتوانیم به این شکل بیان کنیم می‌توانیم از خصوصیاتی که در مورد ترتیب جزئی می‌دانیم در جهت کار با معناشناسی استفاده کنیم.

تعریف ۲.۱. (پیوند گالوا): اگر $(\mathbf{C}, \sqsubseteq)$ و $(\mathbf{A}, \preceq)$ دو ترتیب جزئی باشند و دو تابع $\alpha : \mathbf{C} \rightarrow \mathbf{A}$ و $\gamma : \mathbf{A} \rightarrow \mathbf{C}$ را داشته باشیم که:

$$\forall \mathbf{c} \in \mathbf{C} : \forall \mathbf{a} \in \mathbf{A} : \alpha(\mathbf{c}) \preceq \mathbf{a} \leftrightarrow \mathbf{c} \sqsubseteq \gamma(\mathbf{a})$$

۲.۱ روش وارسی مدل

روش وارسی مدل یک روش صوری است که برای درستی‌یابی سیستم‌های مختلف استفاده می‌شود. در این روش معمولا ابتدا یک ماشین حالات متناهی از روی سیستم مورد بررسی ساخته می‌شود، سپس بررسی‌هایی که قرار است روی سیستم اصلی انجام شوند، روی این ماشین(مدل) انجام می‌شود. در بررسی صحت کارکرد برنامه‌های کامپیوتری از این روش استفاده می‌شود اما این تنها مورد استفاده از این روش نیست و هر منظومه‌ی دیگری که قابلیت بیان به صورت صوری را داشته باشد قابل بررسی با این روش هست. مثلا می‌توان از این روش برای بررسی صحت عملکرد برنامه‌ی قطارهای شهری استفاده کرد؛ در حالتی که مثلا خصوصیات مورد بررسی ما عدم رخ دادن تصادف بین قطارها یا پوشش تمام مناطق شهر باشد. مثال های دیگر استفاده‌ی این روش در علوم کامپیوتر می تواند بررسی صحت عملکرد یک پردازنده یا مثلا الگوریتم تقسیم وظایف یک سیستم عامل باشد. این مثال‌ها هیچ 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.

ایده‌ی روش وارسی مدل 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، با معانی متفاوتی که ممکن است در زبان طبیعی داشته باشند، ساخته می‌شوند. این ادوات غالبا در زبان طبیعی نقش قید را دارند. منطق‌های زمانی بخشی از منطق‌های موجهات هستند که به صوری‌گری ما مفهوم زمان را هم اضافه می‌کنند، یعنی قیدهایی مانند فعلا، بعدا و قبلا. منطقی که در اینجا بیان می‌کنیم LTL نام دارد که یکی از منطق‌های زمانی است که برای روش وارسی مدل استفاده می‌شود. البته در مورد قیدهایی که نام بردیم ذکر این نکته ضروری است که در این بیانی که ما در اینجا از این منطق ارائه داده‌ایم ادوات جدید این فعل‌ها نیستند، هرچند که به کمک ادوات جدید می‌توان ادواتی برای هر یک از این قیود ساخت. این تکه از [۱۸] آورده شده.

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.

۱.۲.۱ زبان LTL

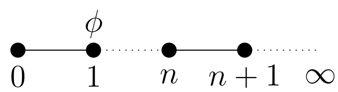
تعریف ۳.۱. هر عضو مجموعه‌ی Φ یک فرمول در زبان LTL است(و Π مجموعه‌ی فرمول‌های اتمی است و $\pi \in \Pi$):

$$\Pi \subset \Phi,$$

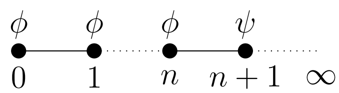
$$\phi \in \Phi \Leftrightarrow \phi ::= \pi|\phi \vee \phi|\neg\phi| \bigcirc \phi|\phi\mathcal{U}\phi$$

اولین نکته‌ای که برای فرمول‌های این زبان به آن نیاز داریم این است که در این منطق ما زمان را با اعداد طبیعی و هر خاصیتی که در موردشان تعریف شده نشان می‌دهیم. یعنی برای یک فرمول، زمان از عدد ۰ شروع شده و تا ابد ادامه خواهد داشت و حین گذر زمان ممکن است ارزش فرمول‌ها تغییر کند. مسلما پس از بررسی معناشناسی صوری بهتر می‌شود این مفهوم را به طور شهودی حس کرد، اما به هر حال به خواننده پیشنهاد می‌شود پیش از رسیدن به آن بخش به ادامه‌ی این بخش هم توجه شود.

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 دقیقا در لحظه‌ی بعدی(دقیقا یک لحظه) است، مثلا در شکل زیر با در نظر گرفتن اینکه در زمان ۰ هستیم، این فرمول در لحظه‌ی ۱ برقرار است.



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



برای این زبان همان‌طور که در منطق کلاسیک از ادوات عطف و شرطی با اینکه با ادواتی که داریم قابل بیان هستند استفاده می‌کنند، ادات بیشتری هم هستند که مورد استفاده قرار می‌گیرند و با همین دو اداتی که معرفی کردیم قابل بیان هستند اما ما در اینجا از آن‌ها اسمی نیاورده‌ایم. دلیل وجود این‌ها هم راحت‌تر کردن کار کسی است که قرار است یک خاصیت را به صورت یک فرمول در این زبان بیان کند. 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. اما از آنجاییکه ما صرفا این بخش را برای معرفی این ایده قرار داده‌ایم، دیگر به بیان ادات وجهی دیگر نپرداخته‌ایم.

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

۲.۲.۱ معناشناسی LTL

مدل‌های این منطق را به صورت توابع $M : \mathbb{N}_0 \to P(\Pi)$ تعریف می‌کنیم. یعنی هر مدل یک تابع است که هر عدد طبیعی را به یک مجموعه از فرمول‌های اتمی می‌برد. این در واقع قرار است به این

معنی باشد که یک مدل به تعبیری به این معنی است که در هر لحظه کدام یک از فرمول‌های اتمی درست هستند. مثلا در مدلی به نام M_i در واقع $M_i(5)$ مجموعه‌ی اتم‌هایی است که در لحظه‌ی ۵ طبق این مدل درست هستند و اگر اتمی در این مجموعه نباشد ارزش غلط دارد. درستی یک فرمول در یک مدل را با M, i نشان می‌دهیم و $M, i \models \phi$ به این معنی است که در لحظه‌ی iدر مدل ϕ ارزش درست دارد. این مفهوم را به صورت بازگشتی به شکل زیر تعریف می‌کنیم:

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

برای یک فرمول اگر مدلی وجود داشته باشد که در آن مدل آن فرمول صادق باشد، آنگاه آن فرمول را ارضاپذیر می‌گوییم. اگر یک فرمول در هر مدلی صادق باشد، آن فرمول را معتبر می‌گوییم. شیوه‌ی دیگری برای بیان همین معناشناسی که گفتیم به شکل اتوماتا است.

فصل ۲

صوری‌گری جدید برای روش وارسی مدل

محوریت کار ما قرار است[۶] باشد که در آن سعی شده روش وارسی مدل با کمک نظریه تعبیر مجرد، بهبود داده شود. در[۴] روشی که معرفی شده درواقع ویژگی برنامه‌ها را به کمک منطق های LTL یا CTL بیان می‌کند. خود برنامه‌ها هم با کمک معناشناسی این منطق ها که نوع خاصی از مدل های کریپیکی به اسم سیستم گذار هستند توصیف می شوند. اما در[۶] کاری که انجام شده به این شکل است که منطق های LTL و CTL با عبارات منظم[۱۵] جایگزین شده‌اند. علت این کار دو نکته بوده، اولی اینکه استفاده از عبارات منظم به جای منطق های نام برده شده می‌تواند برای برنامه نویسان ساده‌تر باشد و دومی اینکه عبارت منظم از از منطق های نام برده شده قدرت بیان بالاتری دارد.[۲۴] در ادامه ی کار وارسی مدل با استفاده از موجودات جدید تعریف شده به سه شکل مختلف بیان شده. در هر مرتبه بیان وارسی مدل، به‌گفته‌ی نویسنده، "ساختارمند"تر شده. می‌توان دریافت که فایده ساختارمندتر بودن بیان این است که پیاده‌سازی راحت‌تری دارد.
حال به بیان و بررسی تعاریف و خواص آن‌ها می‌پردازیم.

۱.۲ نحو زبان مورد بررسی

زبان بیان برنامه ها زیرمجموعه ای از دستورات زبان C است، به شکل زیر:

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

$$\mathsf{A} \in \mathbb{A} ::= 1 \mid \mathsf{x} \mid \mathsf{A}_1 - \mathsf{A}_2$$

$$\mathsf{B} \in \mathbb{B} ::= \mathsf{A}_1 < \mathsf{A}_2 \mid \mathsf{B}_1 \text{ nand } \mathsf{B}_2$$

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

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

$$
\begin{aligned}
& \mathsf{x} \doteq \mathsf{A}; \\
& \mid\ ; \\
& \mid\ \mathsf{if}\ (\mathsf{B})\ \mathsf{S} \mid \mathsf{if}\ (\mathsf{B})\ \mathsf{S}\ \mathsf{else}\ \mathsf{S} \\
& \mid\ \mathsf{while}\ (\mathsf{B})\ \mathsf{S} \mid \mathsf{break}; \\
& \mid\ \{\mathsf{Sl}\} \\
\mathsf{Sl} \in \mathbb{SL} ::= {} & \mathsf{Sl}\ \ \mathsf{S} \mid \epsilon \\
\mathsf{P} \in \mathbb{P}\ ::= {} & \mathsf{Sl}
\end{aligned}
$$

در اینجا زیر مجموعه‌ای از دستورات زبان C را داریم. همین‌طور که قابل مشاهده است این زبان تا حد ممکن کوچک شده. علت این کار را بعدا عمیق‌تر حس خواهیم‌کرد. علتْ ساده‌تر شدن کار برای ارائه‌ی معناشناسی و تعبیر مجرد آن است. در اینجا راحتی آن برنامه‌نویسی که قرار است با این زبان برنامه بنویسد مطرح نبوده چون اصلا این زبان برای این کار ساخته نشده. نویسنده‌ی [۶] در اینجا صرفا می‌خواسته فرآیند را نشان دهد. اگر به فرض برای زبانی مانند پایتون بخواهیم درستی‌یابی با استفاده از روش ارائه شده را درست کینم، می‌توانیم همه‌ی راهی که در [۶] برای زبان توصیف شده، رفته‌شده را برای پایتون هم برویم و به یک تحلیل‌گر ایستا برای پایتون برسیم. در مورد قدرت بیان این زبان هم می‌توانیم بگوییم که می‌توانیم باقی اعداد را از روی عدد ۱ و عملگر منها بسازیم. مثلا ابتدا ۰ را به کمک ۱ـ۱ می سازیم 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 می شوند. باقی اعداد و حتی باقی عملگرها(یعنی به غیر از اعداد طبیعی) نیز از روی آنچه داریم قابل‌ساختن است. در مورد عبارت‌های بولی نیز داستان به همین منوال است. یعنی اینجا صرفا 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 قرار است مطابق چیزی که از زبان C انتظار داریم کار بکنند. در مورد دستور break; ذکر این نکته ضروری است که اجرای آن قرار است اجرای برنامه را از دستوری بعد از داخلی‌ترین حلقه‌ای که break; داخلش قرار دارد ادامه دهد. در پایان می توان ثابت کرد که این زبان هم قدرت با مدل دیویس[۹] است. توجه داریم که هرچه در این بخش درمورد معنای دستورات این زبان گفتیم، به هیچ وجه صوری نبود و صرفا درک شهودی ای که از معنای اجرای هریک از دستورات داشتیم را بیان کردیم. بیان صوری معنای برنامه‌ها را، که برخلاف درک شهودی‌مان قابل انتقال به کامپیوتر است، در ادامه بیان خواهیم‌کرد. طبیعتا این بیان صوری از روی یک درک شهودی ساخته شده‌است.

۲.۲ معناشناسی زبان مورد بررسی

معناشناسی زبانی را که در بخش پیش آوردیم با کمک مفاهیمی به نام برچسب و رد پیشوندی و عملگر چسباندن روی دو رود پیشوندی مختلف تعریف خواهیم‌کرد و نام این معناشناسی نیز معناشناسی رد پیشوندی است.

۱.۲.۲ برچسب‌ها

باوجود اینکه خود زبان C در قسمتی از زبان خود چیزهایی به نام برچسب دارد اما همین‌طور که در بخش پیشین دیدیم، در زبانی که اینجا در حال بحث روی آن هستیم خبری از برچسب‌ها نیست. اما برای تعریف صوری معنای برنامه‌ها، به شکلی که مورد بحث است، به آن‌ها نیاز است. در این بخش ابتدا به توضیحی مختصر در مورد برچسب‌ها در معناشناسی زبان مورد بحث می‌پردازیم. تعاریف صوری دقیق این موجودات در پیوست [۶] آورده‌شده‌اند. از آوردن مستقیم این تعاریف در اینجا خودداری کرده‌ایم. البته در مورد معنای صوری برچسب‌ها هم ذکر این نکته ضروری است که نویسنده‌ی [۶] حتی به صورت صوری هم برای هر بخش از برنامه این کار را به طور دقیق انجام نداده و انجام این کار به طور دقیق‌تر را احتمالاً به کسی که قرار است یک پیاده سازی کامل از این روش داشته باشد سپرده.

در زبانمان Sها بخشی از موجودات موجود در زبان هستند. برچسب ها را برای Sها تعریف می‌کنیم. برچسب‌ها با کمک توابع labs, in, brks-of, brk-to, esc, aft, at تعریف می‌شوند. درواقع هر S به ازای بعضی از این توابع یک برچسب دارد و این‌ها درواقع نشان‌دهنده‌ی آن برچسب هستند. بعض دیگر این توابع برای هر S ممکن است یک مجموعه از برچسب‌ها را تعیین کند و یکی از آن‌ها هم با گرفتن S یک مقدار بولی را برمی‌گرداند.

at[S] : برچسب شروع S.
aft[S] : برچسب پایان S، اگر پایانی داشته باشد.
esc[S] : یک مقدار بولی را بازمی‌گرداند که بسته به اینکه در S دستور ;break وجود دارد یا خیر، مقدار درست یا غلط را برمی‌گرداند.
brk-to[S] : برچسبی است که اگر حین S دستور ;break اجرا شود، برنامه از آن نقطه ادامه پیدا می کند.
brks-of[S] : مجموعه‌ای از برچسب ;break های S را برمی‌گرداند.
in[S] : مجموعه‌ای از تمام برچسب‌های درون S را برمی‌گرداند.
labs[S] : مجموعه‌ای از تمام برچسب‌هایی که با اجرای S قابل دسترسی هستند را برمی‌گرداند.

۲.۲.۲ رد پیشوندی

پس از تعریف برچسب‌ها به سراغ تعریف رد پیشوندی می‌رویم. پیش از آن باید وضعیت‌ها و محیط‌ها را تعریف کنیم.

تعریف ۱.۲. (محیط): به ازای مجموعه مقادیر $\mathbb{V}$ و مجموعه متغیرهای $\mathbb{X}$ تابع $\rho : \mathbb{X} \to \mathbb{V}$ را یک محیط می‌گوییم. مجموعه‌ی همه‌ی محیط‌ها را با $\mathbb{EV}$ نمایش می‌دهیم.

تعریف ۲.۲. (وضعیت): به هر زوج مرتبْ به ترتیب متشکل از یک برچسب l و یک محیط ρ یک وضعیت (یا حالت) $\langle l, \rho \rangle$ می‌گوییم. مجموعه‌ی همه‌ی وضعیت‌ها را با $\mathfrak{S}$ نشان می‌دهیم.

تعریف ۳.۲. (رد پیشوندی): به یک دنباله از وضعیت‌ها(با امکان تهی بودن) یک رد پیشوندی می‌گوییم.

هر رد پیشوندی یک دنباله است که قرار است توصیفی از چگونگی اجرای برنامه باشد. وضعیت‌ها همان‌طور که از نامشان پیداست قرار است موقعیت لحظه‌ای برنامه را توصیف کنند. l قرار است برچسب برنامه‌ی در حال اجرا باشد و ρ مقدار متغیرها را در آن موقع از اجرای برنامه نشان می‌دهد. دنباله‌های ما می‌توانند متناهی یا نامتناهی باشند. مجموعه‌ی ردهای پیشوندی متناهی را با $\mathfrak{S}^{+}$ و مجموعه‌ی ردهای پیشوندی نامتناهی را با $\mathfrak{S}^{\infty}$ نمایش می‌دهیم. مجموعه‌ی همه‌ی ردهای پیشوندی را هم با $\mathfrak{S}^{+\infty}$ نمایش می‌دهیم. باتوجه به آنچه گفتیم، یک عملگر چسباندن $\bowtie$ را روی ردهای پیشوندی تعریف می‌کنیم.

تعریف ۴.۲. (عملگر چسباندن): اگر داشته باشیم $\pi_1, \pi_2 \in \mathfrak{S}^{+\infty}, \sigma_1, \sigma_2 \in \mathfrak{S}$ داریم:

اگر $\pi_1 \in \mathfrak{S}^{+}$ داریم	$\pi_1 \bowtie \pi_2 = \pi_1$
اگر $\sigma_1 \neq \sigma_2$	$\pi_1 \bowtie \pi_2$ تعریف نشده است
اگر $\pi_1 \in \mathfrak{S}^{\infty}$ داریم	$\pi_1\sigma_1 \bowtie \sigma_1\pi_2 = \pi_1\sigma_1\pi_2$

همینطور ϵ هم یک رد پیشوندی است که حاوی هیچ وضعیتی نیست. به عبارت دیگر یک دنباله‌ی تهی است.

۳.۲.۲ تعریف صوری معناشناسی رد پیشوندی

در این بخش قرار است دو تابع $\mathcal{A}$ و $\mathcal{B}$ را به ترتیب روی عبارات حسابی و بولی زبانمان یعنی Aها و Bها تعریف کنیم سپس با کمک آنها S^* را روی مجموعه‌ای از اجتماع معنای Sها و Slها تعریف می کنیم. پس در نهایت هدف ما تعریف S^* است.

تعریف ۵.۲. (معنای عبارات حسابی ـ تابع $\mathcal{A}$): تابع $\mathcal{A} : \mathbb{A} \rightarrow \mathbb{EV} \rightarrow \mathbb{V}$ را به صورت بازگشتی روی ساختار $\mathsf{A} \in \mathbb{A}$ به شکل زیر تعریف می‌کنیم:

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

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

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

تعریف ۶.۲. (معنای عبارات بولی ـ تابع $\mathcal{B}$): تابع $\mathcal{B} : \mathbb{B} \rightarrow \mathbb{EV} \rightarrow \mathbb{BOOL}$ را به صورت بازگشتی روی ساختار $\mathsf{B} \in \mathbb{B}$ به شکل زیر تعریف می‌کنیم:

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

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

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

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

در ادامه به تعریف $\mathcal{S}^*$ می‌پردازیم. این کار را با تعریف $\mathcal{S}^*$ روی هر ساخت S و Sl انجام می‌دهیم. پیش از ادامه‌ی بحث باید این نکته را درمورد علامت‌گذاری‌هایمان ذکر کنیم که منظور از $\mathsf{S} ::= l\mathsf{break};$ این است که تاکید کرده‌ایم که S با برچسب l شروع شده‌است وگرنه همین طور که گفتیم l جزو زبان نیست.

تعریف ۷.۲. (معنای برنامه‌ها ـ تابع $\mathcal{S}^*$): اگر $\mathsf{S} ::= \mathsf{break};$ باشد، ردهای پیشوندی متناظر با اجرای این دستور را به شکل مجموعه‌ی زیر تعریف می کنیم:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

اگر $\mathsf{S} ::= \mathsf{while(B)S_b}$ باشد، ماجرا نسبت به حالات قبل اندکی پیچیده‌تر می‌شود. تابعی به اسم $\mathcal{F}$ را تعریف خواهیم‌کرد که در حقیت دو ورودی دارد. ورودی اول آن یک دستور حلقه است و ورودی دوم آن یک مجموعه. به عبارتی دیگر می‌توانیم بگوییم به ازای هر حلقه یک تابع $\mathcal{F}$ جداگانه تعریف می‌شود که مجموعه‌ای از ردهای پیشوندی را می گیرد و مجموعه‌ای دیگر از همین موجودات را بازمی‌گرداند. کاری که این تابع قرار است انجام دهد این است که انگار یک دور دستورات داخل حلقه را اجرا می کند 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. با دیدن تعریف می توان به دلیل این کار پی‌برد. آن نقطه‌ای که دیگر $\mathcal{F}$ روی آن اثر نمی‌کند یا حالتی است که در آن دیگر شرط حلقه برقرار نیست و اصولا قرار نیست دستورات داخل حلقه اجرا شوند که طبق تعریف $\mathcal{F}$ می‌توانیم ببینیم که $\mathcal{F}$ در این حالت چیزی به ردهای پیشوندی اضافه نمی‌کند. یا اینکه حلقه به دستور ;break خورده 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 است برای اینکه از آن در تعریف صوری معنای حلقه استفاده کنیم. علت اینکه کوچکترین نقطه ثابت

را به عنوان معنای حلقه در نظر می‌گیریم هم این است که مطمئن هستیم کوچکترین نقطه ثابت، هر رد پیشوندی ای را در خود داشته باشد به معنای اجرای برنامه مرتبط است. برای درک بهتر این نکته می‌توان به این نکته توجه کرد که با اضافه کردن وضعیت‌هایی کاملا بی‌ربط به اجرای برنامه به ته ردهای پیشوندی، که صرفا برچسب متفاوتی با آخرین وضعیت هر رد پیشوندی دارند، نقطه ثابت جدیدی ساخته ایم. پس اگر خودمان را محدود به انتخاب کوچکترین نقطه ثابت نکنیم، به توصیفات صوری خوبی از برنامه‌ها دست پیدا نخواهیم‌کرد. در مورد نقطه ثابت تنها این نکته باقی می‌ماند که اصلا از کجا می‌دانیم که چنین نقطه ثابتی وجود دارد که در این صورت باید گفت مجموعه‌هایی که از ردهای پیشوندی تشکیل می‌شوند با عملگر زیرمجموعه بودن یک مشبکه را تشکیل می‌دهند و بنا به قضیه تارسکی[۲۲] برای چنین موجودی نقطه ثابت وجود دارد. تعاریف موجوداتی که درموردشان صحبت کردیم به این شکل است:

$$\mathcal{S}[\mathsf{S}] = lfp^{\subseteq}\,\mathcal{F}[\mathsf{S}]$$
$$\mathcal{F}[\mathsf{S}]X = \{\langle at[\mathsf{S}], \rho\rangle | \rho \in \mathbb{EV}\} \cup$$
$$\{\pi_2\langle l, \rho\rangle\langle aft[\mathsf{S}], \rho\rangle | \pi_2\langle l, \rho\rangle \in X \wedge \mathcal{B}[\mathsf{B}]\rho = False \wedge l = at[\mathsf{S}]\} \cup$$
$$\{\pi_2\langle l, \rho\rangle\langle at[\mathsf{S_b}], \rho\rangle\pi_3 | \pi_2\langle l, \rho\rangle \in X \wedge \mathcal{B}[\mathsf{B}]\rho = True \wedge$$
$$\langle at[\mathsf{S_b}], \rho\rangle\pi_3 \in \mathcal{S}[\mathsf{S_b}] \wedge l = at[\mathsf{S}]\}$$

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

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

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

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

در ادامه چند مثال می‌زنیم تا مفهوم موجودات تعریف شده مشخص‌تر شود.

مثال ۲.۸.

مثال ۲.۹.

مثال ۲.۱۰.

۳.۲ ویژگی‌های معنایی برنامه‌ها

تا به اینجای کار یک زبان آورده‌ایم و برای آن معنا تعریف کرده‌ایم. در این فصل می‌خواهیم در مورد ویژگی‌های برنامه‌هایی که در این زبان نوشته می‌شوند با توجه به معنای صوری‌ای که تعریف کرده‌ایم، صحبت کنیم. دقت شود که برای برنامه‌هایی که در یک زبان برنامه‌نویسی نوشته می‌شوند می‌توان به اشکال مختلفی ویژگی تعریف کرد؛ مثلا ویژگی‌های نحوی، مثل اینکه طول برنامه چند خط است یا هر کاراکتر چند بار به کار رفته، یا ویژگی‌های محاسباتی، مثل بررسی سرعت برنامه یا میزان استفاده‌ی آن از حافظه که عموما در نظریه الگوریتم و پیچیدگی محاسبات بررسی می‌شود. منظور ما در اینجا از تعریف ویژگی، متناسب است با معناشناسی‌ای که برای برنامه‌هایمان تعریف کرده‌ایم. معناشناسی‌ای که تعریف کرده‌ایم درواقع سیر محاسباتی برنامه را توصیف می‌کند و ما می‌خواهیم ویژگی‌ها را با توجه به این موصوع 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.

۱.۳.۲ ویژگی‌های معنایی

همان‌طور که در بخش قبلی دیدیم، معنای هر برنامه با یک مجموعه‌ی $S^*[\mathsf{S}]$ مشخص می‌شود. وقتی می‌خواهیم ویژگی‌هایی را برای موجوداتی که به کمک مجموعه‌ها تعریف شده اند بیان کنیم، اینکه ویژگی‌ها را هم با مجموعه‌ها بیان کنیم کار معقولی به نظر می‌رسد. مثل اینکه بخواهیم ویژگی زوج بودن را در مورد اعداد طبیعی بیان کنیم. می توانیم مجموعه‌ی $\mathbb{E}$ را به عنوان مجموعه‌ی همه‌ی اعداد زوج در نظر بگیریم و اینکه یک عدد زوج هست یا نه را عضویتش در مجموعه‌ی $\mathbb{E}$ تعریف کنیم. پس یعنی در مورد اعداد طبیعی قرار است هر ویژگی به شکل زیرمجموعه‌ای از تمام این اعداد در نظر گرفته شود. 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 $\mathbb{P} \to P(\mathfrak{S}^+)$ است. یعنی یک برنامه را در ورودی می‌گیرد و یک مجموعه از ردهای پیشوندی را باز می‌گرداند. پس می‌توانیم هر ویژگی را به عنوان زیر مجموعه‌ای از $P(\mathfrak{S}^+)$ تعریف کنیم، به عبارت دیگر عضوی از $P(P(\mathfrak{S}^+))$.

۲.۳.۲ عبارات منظم

در اینجا توصیف ویژگی‌ها برای هر برنامه باید یک چارچوب داشته باشد. در صورت قدیمی روش وارسی مدل ما از منطق های 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 وارسی 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.

۳.۳.۲ 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 و عبارت بولی B تشکیل شده‌اند که این زوج را به شکل $\mathsf{L} : \mathsf{B}$ در زبانمان نمایش می‌دهیم.
زبان ما به شکل BNF زیر است:

تعریف ۱۱.۲ .

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

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

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

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

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

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

همان طور که قابل مشاهده است در اینجا عملگرهای دوتایی چسباندن ($\bullet$) و انتخاب ($|$) را داریم، به همراه عملگرهای یگانی * و $^+$. در ادامه خواهیم دید که در فرازبان معنی عملگر یگانی $^+$ به وسیله‌ی عملگر یگانی دیگر قابل بیان است، هرچند که در زبانمان هم برای سهولت کار از بیان این عملگر اجتناب نشده. توجه شود که پرانتزها هم جزئی از زبان قرار داده شده‌اند.

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

با یک نگاه به دستور این زبان یک نکته‌ی چشمگیر برای ما، با توجه به موجوداتی که در بخش قبل تعریف کردیم، با نگاه به قواعد این زبان می‌تواند وجود یک مجموعه‌ی $\underline{\mathbb{X}}$ در کنار $\mathbb{X}$ که از قبل داشتیم باشد. قرار است به ازای هر $\mathsf{x} \in \mathbb{X}$ یک $\underline{\mathsf{x}} \in \underline{\mathbb{X}}$ داشته‌باشیم. منظور از $\underline{\mathsf{x}}$ مقدار متغیر x در ابتدای هر برنامه است. این یعنی تابع $\underline{\rho} : \underline{\mathbb{X}} \to \mathbb{V}$ که $\mathbb{V}$ مجموعه‌ی مقادیر متغیرهاست(در بخش قبل به این اشاره نشد اما خود ρ هایی که در بخش قبل داشتیم هم از نوع $\mathbb{X} \to \mathbb{V}$ بود. با توجه به زبانمان و توضیحاتی که در گذشته دادیم، می‌توان در نظر گرفت که در اینجا $\mathbb{V}$ همان اعداد صحیح است). همان‌طور که پیش‌تر گفتیم برای اشاره به یک تابع ρ از کلمه‌ی "محیط" استفاده می‌شود. به همین منوال در ادامه برای اشاره به $\underline{\rho}$ از "محیط اولیه" استفاده می‌کنیم. برای اشاره به مجموعه‌ی همه‌ی محیط‌های اولیه هم از نماد $\underline{\mathbb{EV}}$ استفاده می‌کنیم. بقیه‌ی موجودات از جمله برچسب‌ها و عبارات بولی را هم که قبلا داشتیم.

در ادامه به بیان صوری معنای زبان بیان شده می‌پردازیم. پس از آن می‌توانیم با بررسی چند مثال، از اینکه معنای هر عبارت منظم چیست درکی شهودی به دست آوریم.

۴.۳.۲ معناشناسی عبارات منظم

معنای عبارات منظم را با استفاده از تابع $\mathcal{S}^r$ نشان می‌دهیم. این تابع به این شکل تعریف می‌شود که در ورودی یک عبارت منظم R را می‌گیرد، سپس یک مجموعه از زوج مرتب‌های(یا همان‌طور که پیش‌تر نام‌گذاری کردیم "وضعیت‌ها"ی) $\langle \underline{\rho}, \pi \rangle$ را که $\pi \in \mathbb{S}^*$ و $\underline{\rho} \in \underline{\mathbb{EV}}$ باز می‌گرداند. بنابراین این تابع از نوع $\mathbb{R} \to P(\underline{\mathbb{EV}} \times \mathbb{S}^*)$ است. همین‌طور دقت شود که تا به حال از $\mathbb{S}^*$ صحبتی نکرده بودیم و فقط $\mathbb{S}^+$ را معرفی کرده بودیم. $\mathbb{S}^*$ نیز برابر است با $\mathbb{S}^+ \cup \{\epsilon\}$ (به لحاظ معنایی همان عملگر $*$ است که در زبان عبارات منظمهم هست، مشهور به ستاره‌ی کلینی).

تعریف استقرایی تابع $\mathcal{S}^r$ به شکل زیر است:

تعریف ۱۲.۲. تابع $\mathcal{S}^r : \mathbb{R} \to P(\underline{\mathbb{EV}} \times \mathbb{S}^*)$ به صورت استقرایی روی ساختار عبارت منظم R به صورت زیر تعریف می‌شود:

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

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

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

[این یعنی معنای عبارت $\mathcal{S}^r[\mathsf{L} : \mathsf{B}]$ زوج مرتب‌هایی هستند که عضو اول آن‌ها محیط‌های اولیه مختلف هستند(مانند مورد قبلی و البته در موارد آتی!) و عضو دوم آن‌ها ردهای پیشوندی تک‌عضوی $\langle l, \rho \rangle$ هستند که در آن‌ها برچسب l باید در L که مجموعه‌ای از برچسب‌هاست حضور داشته باشد و عبارت بولی B باید درباره‌ی محیط اولیه $\underline{\rho}$ و محیط ρ برقرار باشد. حتما متوجه این نکته شدید که $\mathcal{B}$ در اینجا به جای اینکه از نوع $\mathbb{EV} \to \mathbb{BOOL}$ باشد،همان‌طور که قبلا تعریف کردیم، از نوع $\underline{\mathbb{EV}} \to \mathbb{EV} \to \mathbb{BOOL}$ است.(منظور از $\mathbb{BOOL}$ همان مجموعه‌ی $\{True, False\}$ است.) در اینجا $\mathcal{A}$ و $\mathcal{B}$ را در ادامه با نوع‌های متفاوت دوباره تعریف خواهیم کرد، که البته فرق اساسی‌ای با تعریف قبلی ندارد و صرفا گسترشی ساده از آن است.]

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

به‌طوری که در آن برای هر دو مجموعه‌ی $\mathcal{S}$ و $\mathcal{S}'$ از ردهای پیشوندی:

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

[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 با چسباندن ردهای پیشوندی موجود در مولفه‌ی دوم زوج مرتب‌هایی

که اعضای مجموعه‌ی معنای این دو عبارت منظم هستند و گذاشتن این رد پیشوندی‌های حاصل از چسباندنْ در معنای عبارت منظم جدید تعریف می‌شود. همین‌طور که می‌بینید یک عملگر چسباندن برای دو مجموعه از این زوج‌های $\langle \underline{\rho}, \pi \rangle$ تعریف شده و در تعریف $\mathcal{S}^r[\mathsf{R}_1\mathsf{R}_2]$ از آن کمک گرفته شده.
تا این تکه از تعریف معنای عبارت منظم که رسیده‌ایم، تا حدی به دستیابی به درکی شهودی از اینکه به چه نحوی قرار است عبارات منظم راهی برای توصیف ویژگی در مورد برنامه‌ها باشد نزدیک‌تر شده‌ایم. همان‌طور که در مورد قبل دیدیم هر زوج $\mathsf{L} : \mathsf{B}$ دقیقا به یک وضعیتْ 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 منطبق باشند تا وارسی مدل انجام شود. درک این موضوع اولین قدم ماست در دیدن عصاره‌ی روش وارسی مدل در ادبیاتی که از اول این فصلْ عَلَم کرده‌ایم.]

$$\mathcal{S}^r[\mathsf{R}_1 \mid \mathsf{R}_2] = \mathcal{S}^r[R_1] \cup \mathcal{S}^r[R_2]$$

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

$$\mathcal{S}^r[\mathsf{R}]^0 = \mathcal{S}^r[\varepsilon]$$
$$\mathcal{S}^r[\mathsf{R}]^{n+1} = \mathcal{S}^r[\mathsf{R}]^n \bowtie \mathcal{S}^r[\mathsf{R}]$$

[دو عبارت اخیر برای توصیف معنای عملگرهای * و $^+$ تعریف شده‌اند. عملگر $\bowtie$ و معنای $\mathcal{S}^r[\varepsilon]$ را هم که قبلا تعریف کرده بودیم و 0 و n و $n+1$ هم اعداد طبیعی‌اند و $+$ لاجرم همان جمع اعداد طبیعی است.]

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

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

[این دو عبارت هم تعریف معنای خود دو عملگر * و $^+$ هستند. منظور از $\mathbb{N}$ مجموعه‌ی اعداد طبیعی است. همان‌طور که قبل‌تر هم اشاره شد $^+$ را می‌توان در فرازبان با * تعریف کرد. اضافه می‌کنیم که خود * را هم در فرازبان می‌توان با عملگر انتخاب تعریف کرد و در اینجا می‌توان این نکته را هم دید.]

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

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

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

تعریف ۱۳.۲. توابع $\mathcal{A} : \mathbb{A} \to \underline{\mathbb{EV}} \to \mathbb{EV} \to \mathbb{V}$ و $\mathcal{B} : \mathbb{B} \to \underline{\mathbb{EV}} \to \mathbb{EV} \to \mathbb{BOOL}$ به شکل استقرایی به ترتیب روی ساختارهای $\mathsf{A} \in \mathbb{A}$ و $\mathsf{B} \in \mathbb{B}$ به شکل زیر تعریف می‌شوند:

$$\mathcal{A}[1]\underline{\rho}, \rho = 1$$
$$\mathcal{A}[\underline{\mathsf{x}}]\underline{\rho}, \rho = \underline{\rho}(\mathsf{x})$$
$$\mathcal{A}[\mathsf{x}]\underline{\rho}, \rho = \rho(\mathsf{x})$$
$$\mathcal{A}[\mathsf{A}_1 - \mathsf{A}_2]\underline{\rho}, \rho = \mathcal{A}[\mathsf{A}_1]\underline{\rho}, \rho - \mathcal{A}[\mathsf{A}_2]\underline{\rho}, \rho$$
$$\mathcal{B}[\mathsf{A}_1 < \mathsf{A}_2]\underline{\rho}, \rho = \mathcal{A}[\mathsf{A}_1]\underline{\rho}, \rho \ < \ \mathcal{A}[\mathsf{A}_2]\underline{\rho}, \rho$$
$$\mathcal{B}[\mathsf{B}_1 \text{ nand } \mathsf{B}_2]\underline{\rho}, \rho = \mathcal{B}[\mathsf{B}_1]\underline{\rho}, \rho \uparrow \mathcal{B}[\mathsf{B}_2]\underline{\rho}, \rho$$

به‌راحتی قابل مشاهده است که تعاریف جدید تا حد خوبی به تعاریف قبلی شبیه هستند و فرق عمده صرفا وارد شدن $\underline{\rho}$ است.
حال به سراغ چند مثال از عبارات منظم و معنای آن‌ها می‌رویم.

مثال ۱۴.۲. فرض کنید عبارت منظم ما

مثال ۱۵.۲. این سه تا مثال باید با سه تا مثال بخش معناشناسی برنامه‌ها سینک باشن! همین طور در ادامه بعد از تعریف مدل چکینگ در این فصل هم ۳ تا مثال خواهیم داشت که قراره هر کدومشون از این ۳ تا مثال و ۳ تا مثالی که قبلا تعریف کردیم تشکیل شده باشن.

مثال ۱۶.۲.

تا اینجای کار بیشتر مفاهیمی که برای بیان صورت جدید مسئله‌ی وارسی مدل احتیاج داریم را بیان کرده‌ایم.

۵.۳.۲ واریته‌های مختلف زبان عبارات منظم

به عنوان قسمت آخر این بخشْ واریته‌های مختلفی از زبان عبارات منظم را بیان می‌کنیم. ، که هر کدام در واقع زیرمجموعه‌هایی از کل عبارات زبانی که توصیف کرده‌ایم را توصیف می‌کنند. بعضی از آن‌ها را در همین فصل برای هدف نهایی این فصل و بعضی دیگر را در فصل بعدی استفاده می‌کنیم.
اولین واریته‌ای که می‌خواهیم بیان کنیم، واریته‌ای است که در اعضای آن اصلا عبارت $\mathsf{L} : \mathsf{B}$ حضور ندارد و کل عبارت‌های زبان از ε ها تشکیل شده‌اند.

تعریف ۱۷.۲. (عبارت منظم تهی ـ $\mathbb{R}_\varepsilon$):

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

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

با توجه به بخش قبل متوجه هستیم که معنای همه‌ی این عبارت‌ها برابر $\{\langle\underline{\rho}, \epsilon\rangle\}$ خواهد بود. واریته‌ی بعدی عبارت منظم ناتهی است.

تعریف ۱۸.۲. (عبارت منظم ناتهی ـ $\mathbb{R}^+$):

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

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

دلیل وجود $\varepsilon\mathsf{R}_2$ و $\mathsf{R}_1\varepsilon$ در تعریف این است که ممکن است معنای عبارتی با معنای عبارات عضو $\mathbb{R}_\varepsilon$ برابر نباشد(یعنی برابر $\langle\underline{\rho}, \epsilon\rangle$ نباشد)، اما در خود عبارتْ ε حضور داشته باشد. با این تفاصیل می‌توان دید که دو مجموعه‌ی $\mathbb{R}_\varepsilon$ و $\mathbb{R}^+$ یک افراز برای مجموعه‌ی $\mathbb{R}$ هستند، براساس اینکه معنای هر عبارت در $\mathbb{R}$ برابر $\langle\underline{\rho}, \epsilon\rangle$ هست یا خیر. بنابراین شاید به نظر برسد که تعریف یکی از آن‌ها به طور ساختاری کافی بود، اما ممکن است درجایی احتیاج داشته باشیم که ساختاری استقرایی روی هر یک از آن‌ها عَلَم کنیم یا اینکه در اثبات حکمی بخواهیم از استقرا روی یکی از این دو ساختار استفاده کنیم.

واریته‌ی آخر عبارات منظم ما نیز عبارات منظم بدون انتخاب است.

تعریف ۱۹.۲. (عبارت منظم بدون انتخاب ـ $\mathbb{R}^\dagger$):

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

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

۴.۲ صورت جدید مسئله‌ی وارسی مدل

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

پیش از ارائه‌ی تعریف وارسی مدل نیاز داریم تا عملگر بستار پیشوندی را برای یک مجموعه از ردهای پیشوندی معرفی کنیم.

تعریف ۲.۲۰. (بستار پیشوندی): اگر $\Pi \in P(\underline{\mathbb{EV}} \times \mathbb{S}^{+})$ آنگاه بستار پیشوندی Πرا به صورت زیر تعریف می‌کنیم:

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

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

مثال ۲.۲۱. اگر $\Pi = \{\langle \underline{\rho}, \langle l_1, \rho_1\rangle\langle l_2\rho_2\rangle\rangle\langle l_3, \rho_3\rangle\rangle, \langle \underline{\rho}, \langle l_1{'}, \rho_1{'}\rangle\langle l_2{'}\rho_2{'}\rangle\rangle\}$ باشد(Π شامل دو عضو است) آنگاه:

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

که شامل ۵ عضو است.

حال به ارائه‌ی صورت جدیدمان از روش وارسی مدل می‌رسیم که هدف اصلی این اصل بود و با این تعریف فصل تمام می‌شود.

تعریف ۲.۲۲. (وارسی مدل): اگر $\mathsf{P} \in \mathbb{P}, \mathsf{R} \in \mathbb{R}^{+}, \underline{\rho} \in \underline{\mathbb{EV}}$ آنگاه:

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

این تعریف بیان می‌کند که برنامه‌ی P در صورتی که با محیط اولیه‌ی $\underline{\rho}$ اجرا شود، در صورتی خاصیتی که با عبارت منظم R بیان شده را دارد که معنای آن زیرمجموعه‌ی بستار پیشوندی معنای عبارت منظم $\mathsf{R} \bullet (? : T)^{*}$ باشد. توجه شود که محیط اولیه‌ای که برای برنامه‌ی مورد بررسی متصور هستیم صرفا به این منظور قرار داده شده 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 وارسی مدل یعنی فصل ۴ مشخص می‌شود و در این صورت از روش وارسی مدل و صورت بعدی آن چندان نقشی ندارد.

در مورد نقش $(? : T)^{*}$ و prefix این به تصمیم مبدع این روش بوده که این دو در تعریف روش وارسی مدل حضور داشته باشند. حضور این دو طبعا باعث می‌شود به ازای یک عبارت منظم R نسبت به این حالت که صرفا معنی برنامه زیرمجموعه‌ی معنی R باشد، برنامه‌های بیشتری باشند که خاصیت بیان شده با R را ارضا کنند، چون در این صورت مجموعه‌ی سمت راستی در رابطه‌ی زیرمجموعه بودن بزرگتر می‌شود.

۵.۲ در مورد توقف پذیری

در این بخش مشکلی از کار که به نظر نگارنده رسیده مطرح شده. متاسفانه این مشکل بسیار بزرگ است و به نظر عجیب می‌آید اگر نویسنده‌ی [۶] متوجه آن نبوده باشد! 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 ۲۳.۲. (توقف پذیری:) برنامه‌ی P را به همراه اجرای اولیه $\underline{\rho}$ توقف پذیر می‌گوییم اگر و تنها اگر وجود داشته باشد $\pi \in \mathcal{S}^*[\![\mathsf{P}]\!]$ که (ρ محیط متناظر با محیط اولیه‌ی $\underline{\rho}$ است.):

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

و اینکه $\langle aft[\![\mathsf{P}]\!], \rho'\rangle$ در π حضور داشته باشد. این اتفاق را با $\mathsf{P}, \underline{\rho} \downarrow$ نشان می‌دهیم. همین تعریف را برای لیست دستورات Sl یا دستور S هم صرفا با جایگذاری این‌ها با برنامه‌ی P داریم.

در این تعریف توقف پذیری صرفا برای یک محیط اولیه تعریف شده. در اینجا توقف پذیری به متناهی بودن ردهای پیشوندی موجود در برنامه ربط داده نشده. با توجه به معناشناسی‌ای که داریم، تعریف توقف پذیری به معنای وجود رد پیشوندی متناهی با محیط اولیه‌ی مورد بررسی در معنای برنامه که اصلا جور در نمی‌آید، چون معناشناسی ما خاصیت پیشوندی بودن را دارد و مطمئن هستیم در معنای هر برنامه‌ای حتما یک رد پیشوندی متناهی با محیط اولیه‌ی مورد بررسی وجود دارد.

اگر هم بخواهیم تعریف توقف پذیری را وجودنداشتن ردهای پیشوندی نامتناهی با محیط اولیه‌ی مورد بررسی در معنای برنامه در نظر بگیریم در ابتدا به نظر می‌آید که به تعریف قوی‌تری نسبت به آنچه ارائه دادیم رسیده‌ایم. ما در اینجا سعی داریم تعریفی را ارائه کنیم که برای حرف‌هایی که در [۶] زده شده تا حد امکان مشکل درست نکند، که اگر دیدیم با این وجود مشکل وجود دارد مطمئن باشیم که اشتباه در [۶] است و نه حرف ما. پس سعی از ارائه‌ی این تعریف که به نظر از تعریف ارائه شده با کار ناسازگارتر می‌آید اجتناب می‌کنیم(در ادامه به بیان ناسازگاری پراخته شده) اما در قضیه‌ی بعدی می‌بینیم که تعریفی که ارائه کردیم با همان که بگوییم در برنامه رد پیشوندی نامتناهی وجود ندارد معادل است.

قضیه ۲۴.۲. برای برنامه‌ی P و محیط اولیه‌ی $\underline{\rho}$ داریم $\mathsf{P}, \underline{\rho} \downarrow$ اگر و تنها اگر با فرض اینکه ρ محیط متناظر با محیط اولیه‌ی $\underline{\rho}$ است و

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

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

حکم را با استقرا روی ساختار Sl ثابت می‌کنیم.

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

داریم:

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

و طبق تعریف برچسب‌ها داریم:

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

پس حکم برقرار است.

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

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

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

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

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

در معنای دستور به ازای هر ρ وجود دارد که خب در هر صورت این شامل محیط متناظر با $\underline{\rho}$ هم می‌شود .

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

با توجه به معنای این دستوردنباله‌ی زیر در معنای این دستور وجود دارد.

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

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

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

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

در مجموعه‌ی معنای این دستور حضور دارد در حالیکه $\langle at[\![\mathsf{S_t}]\!], \rho\rangle\pi$ داخل معنای $\mathsf{S_t}$ است و طبق فرض استقرا می‌دانیم که برچسب آخرین موقعیت π برابر است با $aft[\![\mathsf{S_t}]\!]$ که طبق تعاریف مربوط

به برچسب‌ها $aft[\![\mathsf{S_t}]\!] = aft[\![\mathsf{S}]\!]$. در صورتی که معنای عبارت بولی غلط باشد هم دنباله‌ی زیر در معنای دستور طبق تعریف موجود است.

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

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

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

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

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

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

در اثبات این سمت قضیه این حالت پیچیده ترین حالت است و در واقع تنها حالتی است که در اثبات آن به فرض قضیه احتیاج داریم! همان طور که پیشتر گفتیم معنای حلقه با استفاده از یک تابع تعریف می‌شود. معنای حلقه کوچکترین نقطه ثابت این تابع است، در حالیکه انگار 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 $\underline{\rho}$ شروع شود در مجموعه‌ی معنای S وجود دارد، چونکه به ازای هر محیط $\acute{\rho}$(نقطه به این خاطر است که با ρ خاص موجود در فرض اشتباه گرفته نشود) حالت $\langle at[\![\mathsf{S}]\!], \acute{\rho}\rangle$ در هر اعمال تابع $\mathcal{F}$ روی هر مجموعه‌ی دلخواه وجود دارد. وقتی معنای S را به عنوان کوچک‌ترین نقطه ثابت $\mathcal{F}$ در نظر گرفته‌ایم پس مطمئن هستیم که آن مجموعه‌ای که کوچکترین نقطه ثابت است شامل رد پیشوندی $\langle at[\![\mathsf{S}]\!], \rho\rangle$ است. این رد پیشوندی با اجرای $\mathcal{F}$ تحت تاثیر قرار می‌گیرد. اگر معنای B در یکی از اعمال های $\mathcal{F}$ غلط باشد، رد پیشوندی $\langle at[\![\mathsf{S}]\!], \rho\rangle\pi\langle aft[\![\mathsf{S}]\!], \rho'\rangle$ در معنای برنامه قرار خواهد گرفت و می‌توانیم بگوییم اجرای دستور با این محیط اولیه توقف پذیر است. می‌دانیم که طبق تعریف تابع به انتهای این رد پیشوندی چیزی اضافه نمی‌شود. از طرف دیگر هم با این محیط اولیه، با توجه به تعریف رد پیشوندی دیگری وجود ندارد که طولانی‌تر از رد پیشوندی مورد اشاره باشد.

در حالت دیگر اگر فرض کنیم هیچ گاه به حالتی نمی‌رسیم که در آن معنای B غلط باشد هم با فرض مسئله به تناقض می‌خوریم، چون در آن صورت تابع $\mathcal{F}$ مدام به طول دنباله‌هایی که با محیط ρ شروع می‌شوند می‌افزاید و این یک دنباله‌ی نامتناهی را خواهد ساخت. در صورتی که معنای B هیچ گاه صحیح نباشد، حداقل حالت $\langle at[\![\mathsf{S_t}]\!], \rho''\rangle$ به ته 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 تفاصیل، این مورد هم ثابت می‌شود.

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

در تعریف تابع aft روی برچسب‌ها در [۶] این تعریف برای این دستور مشخص نیست! در [۷] که در مورد برچسب‌ها بحث شده، نویسنده‌ی [۶] گفته که در مورد آن بخش از تعاریف توابع مربوط به برچسب‌ها که تعریف نشده‌اند برداشت آزاد است و 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 مورد این دستور قرار است برقرار باشد این است که اگر این دستور بخشی از $\mathsf{S_t}$ در حلقه‌ی زیر باشد

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

در این صورت $aft[\![\mathsf{S'}]\!] = brk - to[\![\mathsf{S_t}]\!]$ را طبق تعریف داریم. انتظار می‌رود که $aft[\![\text{break;}]\!] = aft[\![\mathsf{S'}]\!]$ باشد. اینکه دستورات برنامه پس از اجرای ;break از خارج(یا به عبارت بهتر بعد از) حلقه‌ی $\mathsf{S'}$ 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 $\mathsf{S'}$ داخلی ترین حلقه‌ای است که ;break درون آن جای دارد.

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

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

که نشانه‌ی توقف است.

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

در این صورت توقف پذیری $\mathsf{Sl''}$ را از فرض استقرای استقرایی که روی لیست دستورات زده بودیم داریم پس $\{\mathsf{Sl''}\}$ 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 دستورات استقرا می‌زنیم.

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.

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

اگر فرض کنیم این دستور به ازای محیط ρ در حالت اول متوقف شده، در واقع فرض کرده‌ایم در معنای این دستور رد پیشوندی $\langle at[\![\mathsf{S}]\!], \rho\rangle\pi\langle aft[\![\mathsf{S}]\!], \rho'\rangle\pi'$ وجود دارد. باید ثابت کنیم به

ازای π' دلخواه اگر رد پیشوندی $\langle at[\![\mathsf{S}]\!], \rho\rangle\pi\langle aft[\![\mathsf{S}]\!], \rho'\rangle\pi'$ داخل $\mathcal{S}^*[\![\mathsf{S}]\!]$ وجود داشته باشد آنگاه $\pi' = \epsilon$ برقرار است.

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

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

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

طبق تعریف معنای عبارات منظم، هر رد پیشوندی متناهی‌ای داخل مجموعه‌ی سمت راستی رابطه‌ی زیرمجموعه بودن قرار می‌گیرد. این یعنی اگر الگوریتمی برای بررسی $\mathsf{P}, \underline{\rho} \models \mathsf{R}$ داشته باشیم، این الگوریتم می‌تواند تشخیص دهد آیا برنامه‌ی P با محیط اولیه‌ی $\underline{\rho}$ متوقف می‌شود یا خیر! این یعنی الگوریتمی 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، هرچند که پس از رسیدن به بیان دو صورت دیگر خواهیم دید که همین صحبت‌هایی که در مورد این صورت می‌کنیم در مورد صورت‌های دیگر هم بدون در نظر گرفتن معادل بودن این ۳ صورت برقرار است.

فصل ۳

وارسی مدل منظم

در این فصل قرار است به بیانی ساختارمندتر از روش وارسی مدل برسیم. اهمیت ساختارمند تر بودن در این است که بیانی که در فصل پیش داشتیم تا پیاده سازی فاصله‌ی بسیاری دارد، چون همان‌طور که پیش‌تر گفته شد مجموعه‌ها موجودات ساختنی‌ای نیستند و کار با آن‌ها حین نوشتن برنامه‌ای کامپیوتری که قرار است پیاده‌سازی روش مورد بحث ما باشد را سخت می‌کند. ساختاری که در این فصل به صورت روش وارسی مدل اضافه می‌شود، ساختار عبارات منظم است، از این رو پیش از اینکه به بیان وارسی مدل منظم بپردازیم، نیاز داریم که ابتدا به بررسی و تعریف برخی خواص عبارات منظم بپردازیم که در ادامه برای بیان وارسی 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

خیلی ساده 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:

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

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

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

اثبات. برای هر عبارت منظم R داریم:

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

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

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

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

$$\mathcal{S}^r[\![\mathsf{R}_1]\!] = \mathcal{S}^r[\![\mathsf{R}_2]\!] \wedge \mathcal{S}^r[\![\mathsf{R}_2]\!] = \mathcal{S}^r[\![\mathsf{R}_3]\!] \to \mathcal{S}^r[\![\mathsf{R}_1]\!] = \mathcal{S}^r[\![\mathsf{R}_3]\!]$$

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

$\square$

۲.۱.۳ فرم نرمال فصلی

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

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

در تعریف بالا به = دقت شود که با $\Bumpeq$ که در ادامه مورد بحث ماست فرق می‌کند. به سبک رایج 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 که یک فرم نرمال فصلی است و $\mathsf{R} \Bumpeq \mathsf{R}'$ برقرار است. ابتدا این تابع را به صورت استقرایی روی ساختار عبارات منظم تعریف می‌کنیم، سپس خاصیتی که گفتیم را درمورد آن ثابت می‌کنیم. اینْ اثبات این حقیقت خواهد بود که هر عبارت منظم با یک فرم نرمال فصلی هم ارز است.

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

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

$$◀\ \mathsf{dnf}(L : B) = L : B$$

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

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

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

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

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

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

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

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

اثبات. همان طور که گفتیم روی ساختار R استقرا می‌زنیم.

$$▶\ R = \varepsilon :$$

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

که ε یک فرم نرمال فصلی است.

$$▶\ R = L : B :$$

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

که $L : B$ هم یک فرم نرمال فصلی است.

$$▶\ R = R_1R_2 :$$

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

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

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

$\blacktriangleright\ R = R_1 + R_2:$

فرض استقرا این خواهد بود که $\mathsf{dnf}(R_1)$ و $\mathsf{dnf}(R_2)$ ترکیب فصلی نرمال هستند پس $\mathsf{dnf}(R_1 + R_2)$ هم که برابر با $\mathsf{dnf}(R_1) + \mathsf{dnf}(R_2)$ است، ترکیب فصلی نرمال خواهد بود.

$\blacktriangleright\ R = R_1^*:$

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

$$\mathsf{dnf}(R_1^*) = ((R_1^1)^*(R_1^2)^*...(R_1^n)^*)$$

که

$$\mathsf{dnf}(R_1) = R_1^1 + R_1^2 + ... + R_1^n$$

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

$\blacktriangleright\ R = R_1^+:$

طبق چیزهایی که از قبل داریم:

$$\mathsf{dnf}(R_1^+) = \mathsf{dnf}(R_1R_1^*)$$

$$\mathsf{dnf}(R_1^*) = ((R_1^1)^*(R_1^2)^*...(R_1^n)^*)$$

که گیریم $R' = \mathsf{dnf}(R_1^*)$ که عضو $\mathbb{R}^{\dagger}$ است. همین طور فرض می‌کنیم:

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

پس با توجه به تعریف dnf برای عملگر چسباندن خواهیم داشت:

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

$\blacktriangleright\ R = (R_1):$

طبق تعریف داریم:

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

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

$\square$

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

قضیه ۶.۳. برای هر دو عبارت منظم $\mathsf{R}_1, \mathsf{R}_2 \in \mathbb{R}$ داریم:

$$(\mathsf{R}_1 + \mathsf{R}_2)^* \doteqdot (\mathsf{R}_1^* \mathsf{R}_2^*)^*$$

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 متنهاهی‌ای از عبارات منظم اثبات کنیم. در ادامه در واقع از این حکم در اثبات استفاده شده.

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

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

اثبات. طبعا این اثبات با استقرا روی ساختار R انجام می‌شود. توجه شود که در هر حالت از استقرا عبارات منظم $\mathsf{R}_1, \mathsf{R}_2$ در ساختار R حضور دارند، فرض گرفته‌ایم که $\mathsf{dnf}(\mathsf{R}_1) = \mathsf{R}_1^1 + \mathsf{R}_1^2 + \ldots + \mathsf{R}_1^n$ و $\mathsf{dnf}(\mathsf{R}_2) = \mathsf{R}_1^2 + \mathsf{R}_2^2 + \ldots + \mathsf{R}_2^m$.

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

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

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

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

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

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

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

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

$(\supseteq)$:

فرض می‌کنیم $\langle \underline{\rho}, \pi \rangle$ یک عضو دلخواه از $\mathcal{S}^r[\![\mathsf{dnf}(\mathsf{R}_1\mathsf{R}_2)]\!]$ باشد. چون $\mathsf{dnf}(\mathsf{R}_1\mathsf{R}_2) = \Sigma_{\mathsf{i}=1}^{\mathsf{n}_1}\Sigma_{\mathsf{j}=1}^{\mathsf{n}_2}\mathsf{R}_1^{\mathsf{i}}\mathsf{R}_2^{\mathsf{j}}$ پس داریم:

$$\exists k_1, k_2 : \pi \in \mathcal{S}^r[\![\mathsf{R}_1^{\mathsf{k}_1}\mathsf{R}_2^{\mathsf{k}_2}]\!]$$

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

با این وجود داریم:

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

$(\subseteq)$:

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

طبق فرض استقرا داریم:

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

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

(به کمک فرض استقرا)

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

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

(طبق نتیجه‌ای که از قضیه‌ی قبل گرفتیم)

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

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

در اینجا عملگر چسباندن را داریم. در موردهای قبلی این را نشان دادیم که چه‌طور در این حالت حکم برقرار می‌شود. می‌توانیم همان اثبات را درمورد همین عبارت هم ببینیم و بگوییم:

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

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

(طبق فرض استقرا:)

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

□

۳.۱.۳ سر و دم عبارات منظم

در این بخش تعریف تابعی را روی عبارات منظم ارائه می‌کنیم که یک عبارت منظم را می‌گیرد و یک زوج از عبارات منظم را تحویل می‌دهد، سپس به بیان یک قضیه در مورد این تابع می‌پردازیم. این تابع را با fstnxt نشان می‌دهیم. قرار است این تابع یک عبارت منظم را بگیرد و آن را به این شکل تجزیه کند که اولین زوج موجود در عبارت منظم که انگار سر عبارت منظم است، از باقی آن که دم آن عبارت منظم می‌شود، جدا شود. تابع روی عبارات منظم تهی و عبارات منظمی که عملگر $+$ را دارند تعریف نشده.

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

$$\blacktriangleleft\ \mathsf{fstnxt}(\mathsf{L} : \mathsf{B}) = \langle \mathsf{L} : \mathsf{B}, \varepsilon \rangle$$
$$\blacktriangleleft\ \mathsf{fstnxt}(\mathsf{R}_1\mathsf{R}_2) = \mathsf{fstnxt}(\mathsf{R}_2)$$
$$\text{where } \mathsf{R}_1 \in \mathbb{R}_\varepsilon$$

$$\blacktriangleleft\ \mathsf{fstnxt}(\mathsf{R_1R_2}) = (\!|\mathsf{R_1^n} \in \mathbb{R}_\varepsilon\ ?\ \langle \mathsf{R_1^f}, \mathsf{R_2}\rangle : \langle \mathsf{R_1^f}, \mathsf{R_1^n} \bullet \mathsf{R_2}\rangle|\!)$$
$$\text{where } \mathsf{R_1} \notin \mathbb{R}_\varepsilon \text{and } \mathsf{fstnxt}(\mathsf{R_1}) = \langle \mathsf{R_1^f}, \mathsf{R_1^n}\rangle$$
$$\blacktriangleleft\ \mathsf{fstnxt}(\mathsf{R^+}) = (\!|\mathsf{R^n} \in \mathbb{R}_\varepsilon\ ?\ \langle \mathsf{R^f}, \mathsf{R^*}\rangle : \langle \mathsf{R^f}, \mathsf{R^n} \bullet \mathsf{R^*}\rangle|\!)$$
$$\text{where } \mathsf{fstnxt}(\mathsf{R}) = \langle \mathsf{R^f}, \mathsf{R^n}\rangle$$
$$\blacktriangleleft\ \mathsf{fstnxt}((\mathsf{R})) = \mathsf{fstnxt}(\mathsf{R})$$

از این تعریف قرار است در صورتی از وارسی مدل که در این فصل ارائه شده استفاده شود. یک قضیه در آخر این بخش آمده 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.

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

$$\mathsf{R} \quad ::= \quad \mathsf{L : B} \mid \varepsilon \mathsf{R_2} \mid \mathsf{R_1}\varepsilon \mid \mathsf{R_1R_2} \mid \mathsf{R_1^+} | (\mathsf{R_1})$$

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 برای این باید 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:

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

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

$$\blacktriangleright\ \mathsf{R} = \varepsilon\mathsf{R_2}\ :$$

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

$$\blacktriangleright\ \mathsf{R} = \mathsf{R_1}\varepsilon\ :$$

مشابه مورد قبل ثابت می‌شود.

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

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

$$\blacktriangleright\ \mathsf{R} = \mathsf{R_1^+}\ :$$

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

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

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

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

چون $\varepsilon \notin \mathbb{R}^+ \cap \mathbb{R}^\dagger$ پس اصلا این مورد باطل است و در موردش نیاز به اثبات نیست.

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

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

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

اینجا هم با توجه به اینکه طبق فرض استقرا $\mathsf{R}_1, \mathsf{R}_2 \in \mathbb{R}'$ مثل مورد قبل چون $+$ در گرامر $\mathbb{R}'$ حضور دارد، حکم ثابت می‌شود.

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

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

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

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

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

مثل مورد قبلی است.

$\square$

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

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

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 $\mathsf{fstnxt}(R) = \langle L : B, \varepsilon \rangle$. از طرف دیگر $\mathcal{S}^{r}[\![L : B \bullet \varepsilon]\!] = \mathcal{S}^{r}[\![L : B]\!]$ را داریم و ε عضو $\mathbb{R}^{\dagger}$ است.

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

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

$$R = \varepsilon R_2 \mathrel{\hat{\approx}} R_2 \mathrel{\hat{\approx}} L : B \bullet R_2'$$

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

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 $\mathsf{fstnxt}(R_1) = \langle L : B, R_1' \rangle$
در آن صورت بنا بر این که $R_1' \in \mathbb{R}_{\varepsilon}$ برقرار هست یا نه دو حالت را داریم:

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

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

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

در این صورت $\mathsf{fstnxt}(R) = \langle L : B, R_1' \bullet \varepsilon \rangle$ طبق تعریف سر و دم برقرار است. چون $R \in \mathbb{R}^{+} \cap \mathbb{R}^{\dagger}$، پس زیر رشته‌های آن نیز عملگر $+$ را نخواهند داشت، پس $R \in \mathbb{R}^{\dagger}$. در اینجا نیز واضح است که:

$$L : B \bullet R_1' \bullet \varepsilon \mathrel{\hat{\approx}} R_1 \varepsilon = R$$

$$\blacktriangleright\ R = R_1 R_2;$$

اگر یکی از R_1 و R_2 برابر ε باشد که حالات بالا را داریم و اگر هر دو برابر ε باشند هم که اصلا به تناقض می‌خوریم چون در این صورت دیگر در $\mathbb{R}^+$ این عبارت منظم را نداریم. پس تنها یک حالت می‌ماند و آن اینکه هیچ یک از این دو عبارت منظم تهی نباشند. باز هم اگر فرض کنیم $\mathsf{fstnxt}(R_1) = \langle L : B, R'_1 \rangle$ مسئله بنا به اینکه $R'_1 \in \mathbb{R}_\varepsilon$ برقرار هست یا خیر افراز می‌شود، مانند حالت قبل.

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

در این صورت $\mathsf{fstnxt}(R) = \langle L : B, R'_1 \bullet R_2 \rangle$. مانند آنچه بالاتر استدلال کردیم خواهیم داشت $R'_1 \bullet R_2 \in \mathbb{R}^\dagger$. علاوه‌بر این طبق فرض استقرا داریم $L : B \bullet R'_1 \simeq R_1$، پس:

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

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

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

در این صورت $\mathsf{fstnxt}(R) = \langle L : B, R_2 \rangle$. مثل حالت‌های قبل ثابت می‌شود که $R_2 \in \mathbb{R}^\dagger$ و اینکه داریم:

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

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

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

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

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

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

(عبارت بالا فرض استقراست.)

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

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

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

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

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

باز هم با استفاده از فرض استقرا داریم:

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

$$\Rightarrow \mathsf{L} : \mathsf{B} \bullet \mathsf{R}_1' \bullet \mathsf{R}_1^* \doteqdot \mathsf{R}_1\mathsf{R}_1^* \doteqdot \mathsf{R}_1^+$$

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

از فرض استقرا نتیجه می‌شود. □

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

۲.۳ وارسی مدل منظم

همان‌طور که گفتیم در این فصل می‌خواهیم یک صورت معادل از صورتی که در فصل پیش برای روش وارسی مدل آورده شده بود را ارائه کنیم و تا به اینجای فصل صرفا به معرفی چند مفهوم که برای بیان این صورت جدید احتیاج داشتیم پرداختیم. در این یخش ابتدا این صورت جدید را بیان می‌کنیم و سپس به اثبات می‌کنیم که این 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 در حالیکه صورت قبلی ساختاری نداشت.

۱.۲.۳ صورت

در نهایت برای تعریف صورت به یک تابع $\mathcal{M}$ احتیاج داریم که در ورودی‌اش یک زوج متشکل از یک محیط اولیه و یک عبارت منظم را در کنار یک برنامه می‌گیرد و در خروجی همه‌ی ردهای پیشوندی موجود در معنای برنامه را که با عبارت منظم سازگار هستند، داخل یک مجموعه بر می‌گرداند. اما در این بین این مفهوم سازگاری یک رد پیشوندی با یک مجموعه چگونه مشخص می‌شود؟ این نکته‌ای است که تا به حال در مورد آن بحث نکرده‌ایم و الان می‌خواهیم تابع $\mathcal{M}^t$ را با این هدف تعریف کنیم. البته این تابع قرار است یک ویژگی بیشتر هم داشته باشد و این ویژگی

این است که اگر عبارت منظم با رد پیشوندی سازگار نباشد، به ما می‌گوید که از کجای عبارت منظم ناسازگاری وجود داشته واگر هم این دو با هم سازگار باشند این تابع به ما نشان می‌دهد که عبارت منظم تا کجا بررسی شده(قهمیدن این موضوع با نگاه به تعریف ساده‌تر می‌شود و البته نباید فراموش کرد که سازگاری‌ای که داریم، قرار است پیرو صورت قبلی باشد که در آن اگر طول یک عبارت منظم از یک رد پیشوندی بیشتر باشد و صرفا تا اتمام طول رد پیشوندی سازگاری بین این دو برقرار باشد، در نهایت هم سازگار حسابشان می‌کنیم، به عبارت دیگر داریم از نقش عملگر prefix در صورت قبلی صحبت می‌کنیم).

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

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

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

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

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

برای اینکه به $\mathcal{M}$ برسیم به معرفی یک تابع دیگر هم می‌پردازیم. این تابع را با $\mathcal{M}^{\dagger}$ نشان می‌دهیم و 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 $(\underline{\mathbb{EV}} \times \mathbb{R}^{\dagger}) \rightarrow P(\mathfrak{S}^{+\infty}) \rightarrow P(\mathfrak{S}^{+\infty} \times \mathbb{R}^{\dagger})$ می‌گوییم وارسی مدل منظم محدود به $\mathbb{R}^{\dagger}$. ضابطه‌ی این تابع به شکل زیر است:

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

حالا خود $\mathcal{M}$ را تعریف می‌کنیم. تعریف این تابع چیزی نیست جز اجتماع گرفتن از خروجی تابع بالا به ازای عبارات منظمی که در فرم نرمال عبارت منظم ورودی تابع حضور دارند. البته اطلاعاتمان از عبارات منظم در هر زوجی که در خروجی $\mathcal{M}^{\dagger}$ حذف می‌شود، یعنی صرفا ردهای پیشوندی را داریم.

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

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

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

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

می‌توان این طور تصور کرد که به هر حال مجموعه‌ی معنای یک برنامه می‌تواند به مجموعه‌هایی افراز شود که در هر مجموعه ردهای پیشوندی‌ای حضور دارند که محیط اولین وضعیتشان یکسان است. در چنین مجموعه‌ای قاعدتا باید اگر در رد پیشوندی‌هایی که حداقل دو عضو دارند، وضعیت دومشان یکسان است و این رویه برای بقیه‌ی وضعیت‌ها نیز صادق است، یعنی مثلا در رد پیشوندی‌هایی که n عضو وجود دارد، وضعیت nام در همه‌ی آن‌ها یکسان خواهد بود. در هر مجموعه‌ی داخل این افراز یک رد پیشوندی ماکسیمال وجود خواهد داشت که در واقع اجرای کامل برنامه با محیطی که در وضعیت اول قرار دارد است.

حال برای هر برنامه‌ی P که خاصیت R در مورد آن مورد بررسی است، بعد از اینکه $\mathcal{M}\langle \underline{\rho}, \mathsf{R}\rangle \mathcal{S}^*[\![\mathsf{P}]\!]$ محاسبه شد، اگر در هر مجموعه از افرازی که در بالا توصیف کردیم، رد پیشوندی ماکسیمال موجود باشد در آن صورت $\mathcal{M}\langle \underline{\rho}, \mathsf{R}\rangle [\![\mathsf{P}]\!] = [\![\mathsf{P}]\!]$ برقرار خواهد بود. اگر هم این تساوی برقرار نباشد، در افراز $\mathcal{M}\langle \underline{\rho}, \mathsf{R}\rangle [\![\mathsf{P}]\!]$ نسبت به وضعیت‌های متفاوت در اول های پیشوندی، در هر افزاری که رد پیشوندی ماکسیمال موجود در همان افراز در مجموعه‌ی $\mathcal{S}^*[\![\mathsf{P}]\!]$ نیست، آن رد پیشوندی دیگری که در آن مجموعه ماکسیمال است، به ما می‌گوید که برنامه به چه وصعیتی برخورد کرده که با R ناسازگاری داشته و اطلاعاتی را در مورد برنامه به ما می‌دهد.

برای حرف هایی که در چند بخش قبل گفتیم، اثباتی نیاورده‌ایم و البته که ادعاهایی هم هستند که نیاز به اثبات دارند، اما به هر حال اثبات آن‌ها در این بخش و در حین بحثی که هستیم، به نظر منحرف کننده می‌آید و البته در ادامه هم حرفی بر اساس این ادعاها زده نشده. اما به هر حال اگر یک درک شهودی از این موجوداتی که در اینجا توصیف شده‌اند داشته باشیم، می‌توانیم تا حد قابل قبولی صحت این ادعاها را ببینیم.

فصل ۴
وارسی مدل ساختارمند

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 boord phase i report., November 1999.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[24] 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