زبان توصیف رسمی Z

سودابه محمدی

عضو هیأت علمی دانشگاه صنعتی کرمانشاه

انتشارات دانشگاه صنعتی کرمانشاه

۱۳۹۸

سرشناسه	: محمدی، سودابه، ۱۳۰۰
عنوان	: زبان توصیف رسمی z
مشخصات نشر	: کرمانشاه، دانشگاه صنعتی کرمانشاه، ۱۳۹۸=۲۰۱۹ م.
مشخصات ظاهری	: ع+۳۷۳ ص. مصور، جدول.
فروست	: دانشگاه صنعتی کرمانشاه؛ ۲۳۶
شابک	: ۹۷۸-۶۰۰-۰۰۰۰-۰۰-۹
یادداشت	: پشت جلد به انگلیسی: Calculus and Analytic Geometry
یادداشت	: کتاب‌نامه
یادداشت	: نمایه
موضوع	: علوم پایه
شناسه افزوده	: دانشگاه صنعتی کرمانشاه
رده‌بندی کنگره	: ۱۳۹۳ ۵آ ۶ ن / XX۸۰۰/۲
رده‌بندی دیویی	: ۵۰۰/۵

لوگو
دانشگاه فلان

انتشارات دانشگاه صنعتی کرمانشاه ۱۱۱

عنوان کتاب: زبان توصیف رسمی Z

تألیف: سودابه محمدی

ویراستار ادبی: علی جبرائیلی

صفحه‌آرا: وحید دامن‌افشان

ناشر: دانشگاه کرمانشاه

تاریخ و نوبت چاپ: ۱۳۹۸ ـ پنجم

شمارگان: ۱۰۰۰

قیمت: ۲۷۰۰۰ تومان

شابک: ۹۷۸-۶۰۰-۰۰۰۰-۰۰-۹

قطع: وزیری

چاپخانه: زلال

مراکز پخش: کتابیران، دانشیران

مسئولیت درستی مطالب به عهده نویسنده است.

تقدیم به همه آن‌هایی که می‌خواهند بیشتر بدانند...

پیش‌گفتار

با توجه به کاربرد و اهمیت روزافزون ریاضیات عمومی در کمک به درک و توجیه پدیده‌های علمی و نیز نظر به اینکه کتاب‌های ریاضی‌ای که تاکنون به زبان فارسی در رابطه با موضوع ریاضیات عمومی ترجمه یا تالیف شده است، نیازهای فعلی جامعه ریاضی و علمی را برآورده نمی‌کند، تصمیم به تالیف کتاب حاضر گرفته شد.

سطح این کتاب به گونه‌ای است که برای دانشجویان سال اول دوره کارشناسی رشته ریاضی و دانشجویان کارشناسی رشته‌های فیزیک، مکانیک و سایر رشته‌های مرتبط قابل استفاده است.

از ویژگی‌های این کتاب، توجه به سرفصل‌های درس نظریه ریاضیات همومی در دوره کارشناسی است؛ به گونه‌ای که تمامی سرفصل‌های مصوب وزارت علوم، تحقیقات و فناوری با بیانی ساده و قابل فهم آورده شده است. همچنین، با توجه به تعدد مثال‌ها، کتاب، به صورت خودخوان نیز قابل استفاده است.

کتاب حاضر از شش فصل تشکیل شده است. در فصل اول، مفاهیم و مقدمات اولیه مورد بررسی قرار گرفته و نیز قضیه اساسی وجودی و منحصر بفردی جواب بیان شده است.

در فصل دوم، مباحث و مطالب فصل اول، روی سیستم معادلات دیفرانسیل مرتبه اول، توسیع داده شده است. همچنین در این فصل، سه روش مختلف برای حل سیستم معادلات ارایه

شده است. لازم به ذکر است که روش حل سیستم معادلات با استفاده از روش جردن، بیشتر برای دوره‌های کارشناسی ارشد آورده شده است. لذا برای دوره‌های کارشناسی می‌توان از مطالعه این روش، چشم‌پوشی کرد. در ادامه فصل، معادلات دیفرانسیل مرتبه nام و قضیه‌های مربوط به آن، بررسی شده است.

فصل سوم در ارتباط با مسایل مقدار مرزی و نظریه اشتورم است. در این فصل، قضیه‌های اساسی در ارتباط با مسایل مقدار مرزی، از جمله قضیه مقایسه‌ای و قضیه تفکیک آورده شده است.

در فصل چهارم، سیستم‌های دینامیکی معرفی شده است. تعاریف و مفاهیم نقاط ثابت، پایداری نقاط ثابت و تصویر فاز، با بیانی ساده و روان ارایه شده است.

فصل پنجم، درباره سیستم‌های دینامیکی خطی در صفحه بحث می‌کند. به بیان دقیق‌تر، سیستم‌های خطی متعارف و سیستم‌های خطی ساده در صفحه، بیان و تصاویر فاز مربوط به آن‌ها مورد کاوش قرار گرفته است.

فصل ششم درباره سیستم‌های غیرخطی در صفحه است. در واقع این فصل، دربرگیرنده مطالب تکمیلی فصل پنجم است. بیشتر مطالب این فصل، برای دوره‌های تحصیلات تکمیلی مناسب است.

از ویژگی‌های این کتاب، توجه به سرفصل‌های درس نظریه ریاضیات همومی در دوره کارشناسی است؛ به گونه‌ای که تمامی سرفصل‌های مصوب وزارت علوم، تحقیقات و فناوری با بیانی ساده و قابل فهم آورده شده است. همچنین، با توجه به تعدد مثال‌ها، کتاب، به صورت خودخوان نیز قابل استفاده است.

امید است که خوانندگان گرامی، نظرها و پیشنهادهای خود را با ما در میان گذاشته تا در چاپ‌های بعدی موجب غنی‌تر شدن کتاب گردد.

وحید دامن‌افشان

کرمانشاه، تابستان ۹۸

فهرست مطالب

۱

مقدمه ای بر توصیف رسمی برنامه ها و سیستم ها

امروزه به همراه هر نرم افزار و یا سیستمی، مجموعه وسیعی از مستندات نیز ارائه می شود. این مستندات شامل : راهنمای کابر، کتاب راهنمای مرجع [1] ، سیستم های راهنمای آنلاین، آموختارهای تعاملی [1] و مستندات طراحی است. با این حال، رفتار نرم افزار، همچنان برای کابران و طراحان، غافل گیر کننده می باشد. گاها مولفه های برنامه به درستی عمل نکرده و سیستم در مواجهه با نیازمندی های کاربر، با شکست مواجه می شود.

در علوم کامپیوتر، توصیفات رسمی، تکنیک‌های مبتنی بر ریاضیات هستند که هدف آنها کمک به پیاده‌سازی سیستم‌ها و نرم‌افزارها است. توصیف ها برای شرح چگونگی عملکرد یک سیستم، تحلیل رفتار سیستم و بررسی و تایید مشخصات کلیدی آن بکار برده می شوند. این توصیفات رسمی هستند به این معنا که دارای یک نحو هستند، از لحاظ معنایی در یک دامنه قرار می گیرد و می توان از آنها برای درک و دریافت اطلاعات مفید استفاده کرد.

[1]reference manuals [1]interactive tutorials

تحلیل نیازمندی ها و توصیف آنها، مبتنی بر ارتباط بین کارفرما، کاربران و تحلیل گران و توسعه دهندگان سیستم است. روش های معمول تحلیل و طراحی سیستم های نرم افزاری، به میزان زیادی، متکی بر زبان های طبیعی و نمادهای گرافیکی است. به این ترتیب ممکن است در توصیف یک سیستم نرم افزاری مشکلات زیر رخ دهند:

۱. تناقض: بیان های متفاوت از موضوعی واحد در قسمت های مختلف مستند نیازمندی ها. مثلا در یک قسمت پایش دما در تمامی حالات خواسته شده است و در بخشی دیگر، در محدوده ای خاص

۲. ابهام: امکان وجود برداشت های مختلف از یک عبارت خاص. (مثلا در جایی که در مستند نیازمندی ها پراگراف زیر نوشته شده است، مشخص نیست که جمله آخر درخصوص گذرواژه است یا شناسه کابر.

"شناسایی کاربر بوسیله نام کاربری و گذرواژه صورت می گیرد. که باید ترکیبی از حروف و ارقام باشد.

۳. غیردقیق بودن: غیردقیق بودن به این معناست که در بیان نیازمندی ها، عبارات کلی گفته شود. (به عنوان مثال در عبارت " فاصل کاربری، باید کاربرپسند باشد" دقیقا مشخص نشده که منظور از کاربر پسند بودن چیست و به بیان این نیاز، بصورت کلی بسنده کرده است.)

۴. کامل نبودن

روش های رسمی از منطق و ریاضیات ساده برای بیان نیازمندی های یک سیستم یا نرم افزار استفاده می کنند. همچنین در توصیف های رسمی از فرمول ها، نمادها و قواعد، استفاده می شود. با استفاده از توصیف رسمی، درک بهتر از سیستم در حین تحلیل سیستم فراهم می شود در حالیکه در سیستم های فاقد توصیف رسمی، این درک و دریافت، بعد از ساخت و با تست سیستم، فراهم می شود. در واقع یکی از اهداف اصلی توصیف های رسمی، آشکارسازی خطا در زمان تحلیل و نه در زمان تست و بعد از ساخت و تجربه کاربر است. در این روش ها، تحلیل بصورت خودکار و با استفاده از ماشین، انجام می شود.از جمله ایرادات توصیف های رسمی می توان به موارد زیر اشاره کرد:

۱. در کنترل پروژه کاربرد ندارند.

```
def f(a):
    term=1
    sum=1
    i=0
    while(sum<=a):
        term=term+2
        sum=sum+term
        print("the sum is: %f and the term is %f",sum,term)
        i=i+1
    return i
```

شکل ۱.۱ تعریف تابع در پایتون

۲. مستندات آن برای مشتری، مفید و قابل درک نیست.

۳. برای پروژه بصورت هزینه سربار است.

۴. نباید آن را به عنوان جایگزینی برای تست ها درنظر گرفت

۱.۱ اولین مثال از توصیف رسمی یک برنامه

به تصویر ۱.۱ که یک برنامه ساده در زبان پایتون است توجه کنید. این برنامه چه کاری انجام می دهد؟

با توجه به تعریف تابع $f(۴) = ۲$ ، اما اگر $a = ۱۰$ و یا $a = -۱۰$ باشد، آنگا تابع چه مقداری را برمی گرداند.

برای درک بهتر کد پایتون شکل ۱.۱ می توان نام تابع را به $iroot$ تغییر داد و یک خط توضیحات به آن اضافه کرد (شکل ۲.۱).

در اینجا متوجه می شویم که در توضیح این تابع، به چیزی بیشتر از کد نیاز داریم. کد، نحوه محاسبه خروجی را نشان می دهد در حالیکه "توصیف" نتیجه محاسبه را دربردارد. توصیف کد شکل ۱.۱ به زبان توصیف رسمی، Z، به شکل ۳.۱ خواهد بود.

از توصیف رسمی، نمی توان به کد رسید. در توصیف برنامه، مشخص می شود که برنامه چه کاری انجام خواهد داد و در واقع به پرسش $what$ پاسخ داده خواهد شد. اما کد پایتون برنامه مشخص می کند که چگونه این کار انجام می شود و پاسخ به پرسش How را دربردارد. این دو

```
def iroot(a): #integer square root
    term=1
    sum=1
    i=0
    while(sum<=a):
        term=term+2
        sum=sum+term
        print("the sum is: %f and the term is %f",sum,term)
        i=i+1
    return i
```

شکل ۲.۱ تعریف تابع در پایتون به همراه توضیحات بیشتر

$$
\begin{array}{|l}
\textit{iroot} \\
\hline
\textit{iroot} : \mathbb{N} \longrightarrow \mathbb{N} \\
\hline
\forall a : \mathbb{N} \\
\textit{iroot}(a) * \textit{iroot}(a) \leq a < (\textit{iroot}(a) + ۱) * (\textit{iroot}(a) + ۱) \\
\hline
\end{array}
$$

شکل ۳.۱ توصیف کد پایتون بوسیله زبان توصیف رسمی Z

تکمیل کننده یکدیگرند و در طراحی یک سیستم و یا یک نرم افزار، به هر دو نیاز داریم. از مثال مطرح شده می توان به این نتیجه رسید که زبان های توصیف رسمی، زبان برنامه نویسی نیستند به این معنا که برای آنها کامپایلری که بتواند کد قابل اجرا تولید کند، وجود ندارد.

۲
معرفی Z

Z یک زبان توصیف رسمی مدل گراست که در دهه ۸۰ توسط گروه پژوهشی برنامه نویسی دانشگاه آکسفورد، توسعه داده شد.این زبان مبتنی بر تئوری مجموعه Zermelo است. Z در سال ۲۰۰۲، استاندارد ISO را دریافت کرد.از آن زمان تاکنون، Z در طیف گسترده ای از نرم افزارهای سیستمی مانند سیستم های پایگاه داده، سیستم های تراکنشی، سیستم های محاسبات توزیع شده و سیستم عامل ها بکار برده شده است. موفقیت قابل توجه Z در توصیف فاصل برنامه نویسی کاربردی CICS بود که بوسیله آزمایشگاه IBM در پارک Hursley انجام شد. تقریبا ۳۷۰۰ خط کد توسط Z تولید شد. این پروژه یک پروژه صنعتی بود که توصیف آن توسط Z منجر به کاهش ۲.۵ درصدی خطا نسبت به حالتی که توصیف Z وجود نداشته باشد، گردید. توصیف های Z ریاضیاتی هستند و از منطق دومقداری استفاده می کنند. استفاده از ریاضیات، صحت این زبان را تضمین می کند و به شناسایی تناقضات موجود در توصیف ها، کمک می کند. Z یک رویکرد مدل گرا است که یک مدل صریح از حالت ماشین انتزاعی را نشان می دهد. عملگرها در این حالت تعریف شده اند. ریاضیات، در Z برای نشان گذاری توصیف های رسمی و حساب شِما، برای ساختار این توصیف ها بکار می روند. شِماها از نظر بصری قابل توجه هستند و بخش اصلی آنها شامل جعبه هایی است. شِماها برای توصیف حالات و عملیات

$$
\begin{array}{|l}
\multicolumn{1}{l}{\underline{\quad SqRoot \qquad\qquad\qquad\qquad\qquad\qquad}} \\
num?, root! : \mathbb{R} \\
\hline
num? \geq ۰ \\
root!^{۲} = num? \\
root! \geq ۰ \\
\hline
\end{array}
$$

شکل ۱.۲ توصیف تابع جذر

بکار می روند. حساب شِما، به شِماها این قابلیت را می دهد که با دیگر شِماها ترکیب شوند و یا در کنار هم بلوک ها را بسازند.تصویر ۱ ، یک شِمای ساده را توصیف می کند. این شِما، توصیف ریشه مرتبه دوم مثبت یک عدد حقیقی است.

عملگرهای شِما بصورت پیش شرط/پس شرط تعریف می شوند. یک پیش شرط باید قبل از اینکه عملگر اجرا شود، بررسی شود در حالیکه پس شرط، پس از اجزای عملگر بررسی می شود. مسلما پس از اثبات درستی پیش شرط ها، عملگر اجرا خواهد شد.پیش شرط بصورت ضمنی در داخل عملگر تعریف شده است. هر عملگر یک فرض اثبات شده به همراه دارد که تضمین می کند درصورت درست بودن پیش شرط، عملگر، تغییرناپذیری سیستم را حفظ کند. تغییرناپذیری یکی از ویژگی های سیستم است که همواره و در هر زمانی باید درست باشد. حالت اولیه سیستم نیز بنحوی است که تغیرناپذیری سیستم برآورده شود. در شکل ۱.۲ ، پیش شرط توصیف تابع جذر بصورت $num? \geq ۰$ است. بنابراین تابع $SqRoot$ تنها ریشه اعداد حقیقی مثبت را بدست می آورد. پس شرط های تابع جذر $root!^{۲} = num?$ و $root! \geq ۰$ هستند. این دو شرط بیان می کنند که اولا ریشه عددی مثبت است و ثانیا توان دو ریشه برابر با عدد ورودی است.

Z یک زبان مبتنی بر نوع است ، به این معنا که وقتی متغیری معرفی می شود، باید نوع آن نیز مشخص گردد. یک نوع، مجموعه ای از اشیا است. چندین نوع استاندارد در Z وجود دارند. این انواع عبارتند از اعداد طبیعی $\mathbb{N}$، اعداد صحیح $\mathbb{Z}$، و اعداد حقیقی $\mathbb{R}$. اعلان یک متغیر به نام x که از نوع X است، بصورت x:X انجام می گیرد.همچنین در Z امکان تعریف نوع توسط

$$
\begin{array}{|l}
\multicolumn{1}{l}{\underline{\quad Library \quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad}} \\
on\_shelf, missing, borrowed : \mathbb{P} Bkd\_Id \\
\hline
on\_shelf \cap missing = \oslash \\
on\_shelf \cap borrowed = \oslash \\
borrowed \cap missing = \oslash \\
\hline
\end{array}
$$

شکل ۲.۲ توصیف یک سیستم کتابخانه

برنامه نویس نیز وجود دارد.

در توصیف های Z از قراردادهای مختلفی استفاده می شود، برای مثال $v?$ بیان کننده این است که v یک متغیر ورودی است و $v!$ بیانگر این است که v، یک متغیر خروجی است. در تابع $SqRoot$ که در بالا تعریف شد، $num?$ یک متغیر ورودی، و $root!$ یک متغیر خروجی را اعلان می کند. علامت Ξ در یک شِما، نشاندهنده این است که عملگر، حالت را تغییر نمی دهد، درحالیکه علامت Δ بیانگر این است که عملگر، باعث تغییر حالت می گردد.

بسیاری از انواع داده های مورد استفاده در Z، مشابهی در زبان های برنامه نویسی استاندارد ندارند. بنابراین لازم است که ساختمان داده های توافقی، شناسایی و توصیف شوند تا درنهایت برای نمایش ساختارهای ریاضیاتی انتزاعی بکار روند. باتوجه به اینکه ساختارهای توافقی ممکن است با انتزاع تفاوت داشته باشند، عملگرهای مربوط به ساختار داده های انتزاعی، نیازمند پالایش به عملگرهای داده های توافقی هستند. این پالایش باعث می شود که نتایج حاصل، یکسان گردد. برای سیستم های ساده، پالایش مستقیم امکان پذیر است. برای بیشتر سیستم های پیچیده، پالایش با تاخیر، بکار برده می شود که در آن یک دنباله از توصیف های توافقی افزاینده، برای توصیف های قابل اجرا، تولید می شوند.

تصویر ۱.۳ نشاندهنده توصیف Z برای امانت گرفتن کتاب از یک سیستم کتابخانه است. کتابخانه شامل کتاب هایی است که در قفسه قرار دارند، کتاب هایی که به امانت رفته اند و کتاب هایی که گم شده اند. توصیف، با استفاده از مجموعه هایی که نشان دهنده کتاب های موجود در قفسه، به امانت رفته و گمشده است، کتابخانه را مدلسازی می کند. بنابراین سه

$$
\begin{array}{|l}
\multicolumn{1}{l}{\underline{\ Borrow\ }}\\
\Delta Library\\
b? : Bkd\_Id\\
\hline
b? \in on\_shelf = on\_shelf \setminus \{b?\}\\
borrowed' = borrowed \cup \{b?\}\\
\hline
\end{array}
$$

شکل ۳.۲ توصیف عملگر امانت گرفتن کتاب

زیرمجموعه مجزا از مجموعه کتاب ها وجود دارد. $Bkd - Id$ شناسه هر کدام را مشخص می کند.

وضعیت سیستم با استفاده از شِمای $Library$ در شکل ۲.۲ مشخص شده است. دو عمل $Borrow$ و $Return$ می توانند بر روی حالت سیستم تاثیر بگذارند. عملگر $Borrow$ در شکل ۲ ـ ۱ توصیف شده است. نشانگذاری $\mathbb{P}Bkd - Id$ مجموعه توانی $Bkd - Id$ (مجموعه تمام زیرمجموعه های $Bkd - Id$) را نشان می دهد. شرط مجزا بودن سه زیرمجموعه on-shelf، missing و borrowed در شِمای $Library$ تعریف شده است. این شرط با تهی بودن اشتراک دو به دوی این مجموعه ها مشخص شده است. پیش شرط عمل $Borrow$ (امانت دادن) این است که کتاب در قفسه موجود باشد. پس شرط آن این است که کتاب به مجموعه کتاب های به امانت رفته اضافه شود و از مجموعه کتاب های موجود در قفسه، حذف شود.

۳
عناصر Z

۱.۳ مجموعه ها

یک مجموعه، دسته ای از اشیا خوش تعریف است. مجموعه ها گاهی با لیستی از تمامی عناصرشان نشان داده می شوند. به عنوان مثال مجموعه اعداد طبیعی زوج کوچتر یا مساوی ۱۰، برابر است با

$$\{۲, ۴, ۶, ۸, ۱۰\}$$

مجموعه ها ممکن است با بکارگیری برخی از عملیات بر روی دیگر مجموعه ها ایجاد شوند. برای مثال مجموعه اعداد طبیعی زوج کوچکتر مساوی ۱۰، با استفاده از عملگرهای مجموعه، بصورت زیر تعربف می شود:

$$\{n : \mathbb{N} \mid n \neq ۰ \wedge n < ۱۰ \wedge n mod ۲ = ۰ \bullet n\}$$

تعریف مجموعه فوق شامل سه بخش است. بخش اول، امضای مجموعه است که بصورت $n : \mathbb{N}$ نشان داده شده است. بخش اول با استفاده از خط عمودی | از بخش دوم جدا می شود. بخش دوم با استفاده از یک شرط بیان بیان می شود. در مثال فوق این شرط عبارت است از

$n \neq o \wedge n < ۱۰ \wedge nmod۲ = ۰$. بخش دوم با استفاده از • از بخش سوم جدا شده است. بخش سوم یک عبارت است که در مثال فوق این عبارت n می باشد. این عبارت ممکن است عبارت پیچیده تری باشد مانند $\log(n^۲)$.

در ریاضیات تنها یک مجموعه تهی وجود دارد. در Z یک مجموعه تهی برای هر نوع از مجموعه ها وجود دارد. ازاینرو به تعداد نامتناهی مجموعه تهی در Z وجود دارد. مجموعه تهی بصورت $\varnothing[X]$ تعریف می شود که در آن X نوع مجموعه تهی را نشان می دهد. اگر نوع واضح باشد، نیازی به نوشتن X نیست.

در Z عملگرهای متنوعی برای مجموعه ها وجود دارد مانند اجتماع، اشتراک، تفاوت مجموعه ها و تفاوت متقارن. مجموعه توانی یک مجموعه مانند X، شامل تمام زیرمجموعه های مجموعه X است و با $\mathbb{P}X$ نشان داده می شود. مجموعه زیرمجموعه های غیرتهی X با $\mathbb{P}_۱X$ نشان داده می شود که در آن

$$\mathbb{P}_۱ == \{U : \mathbb{P}X \mid U \neq \varnothing[X]\}$$

یک مجموعه متناهی از عناصر نوع X که با $\mathbf{F}X$ نشان داده می شود، زیرمجموعه ای از X است که نمی تواند یک تناظر یک به یک با زیرمجموعه خاصی از خودش داشته باشد. تعریف $\mathbf{F}X$ بصورت زیر است.

$$\mathbf{F}X == \{U : \mathbb{P}X \mid \neg \exists V : \mathbb{P}U \bullet V \neq U \wedge (\exists f : V \rightarrow U)\}$$

عبارت $f : V \longmapsto U$ بیان می دارد که f یک رابطه یک به یک از U به V است که در آن هر عضو از مجموعه U دقیقا به یک عضو از مجموعه V نگاشت می شود و برعکس.

Z یک زبان نوع دار است به این معنا که متغیر در هنگام تعریفش، برای اولین بار اعلان می شود. تعریف متغیر با استفاده از سورهای عمومی و وجودی انجام می گردد. برای مثال

$$\forall j : J \bullet P \Rightarrow Q$$

.

کمیت وجودی یکتا بصورت $\exists_۱ j : J \mid P$ تعریف می شود. این تعریف بیان می کند که دقیقا یک j از نوع J وجود دارد که دارای ویژگی P است.

۲.۳ رابطه ها

رابطه R میان X و Y زیرمجموعه ای از ضرب کارتزین X و Y است؛ یعنی $R \subseteq (X \times Y)$.
رابطه در Z بصورت $R : X \longleftrightarrow Y$ نمایش داده می شود. رابطه $x \mapsto y$ نشاندهنده این است که زوج $(x, y) \in R$.

توجه کنید که رابطه $home\_owner : Person \longleftrightarrow Home$ بین افراد و خانه هایشان برقرار است. رابطه $daphne \longmapsto mandalay \in home\_owner$ بیان می دارد که *daphne* مالک *mandalay* است. همچنین امکان دارد که یک شخص بیش از یک خانه داشته باشد:

$rebecca \longmapsto nirvana \in home\_owner$
$rebecca \longmapsto tivoli \in home\_owner$

همچنین ممکن است دو نفر بصورت مشترک مالک خانه ای باشند:

$rebecca \longmapsto nirvana \in home\_owner$
$lawrence \longmapsto nirvana \in home\_owner$

ممکن است افرادی وجود داشته باشند که مالک هیچ خانه ای نیستند. بنابراین، برای آنها، ورودی در رابطه *home_owner* وجود ندارد. نوع *Person* شامل هر فرد ممکنی است و نوع *Home* دربرگیرنده هر خانه ممکنی می باشد. دامنه رابطه *home_owner* بصورت زیر تعریف می شود:

$$x \in \mathrm{dom}\ \ home\_owner \Leftrightarrow \exists h : Home . x \longmapsto h \in home\_owner$$

برد رابطه *home_owner* را نیز می توان بصورت زیر تعریف کرد:

$$h \in \mathrm{ran}\ \ home\_owner \Leftrightarrow \exists x : Person . x \longmapsto h \in home\_owner$$

ترکیب دو رابطه $home\_owner : Person \leftrightarrow Home$ و $home\_value : Home \leftrightarrow Value$ منجر به رابطه $owner\_wealth : Person \leftrightarrow value$ می شود. این رابطه با ترکیب رابطه ای[۱] $home\_owner;\ home\_value$ نشان داده می شود که در آن

$$p \mapsto v \in home\_owner;\ home\_value \Leftrightarrow$$
$$(\exists h : Home . p \mapsto h \in home\_owner \land h \mapsto v \in home\_value)$$

---

[۱] composition relational

ترکیب رابطه ای همچنین ممکن است بصورت زیر نشان داده شود:

$$owner\_wealth = home\_value \circ home\_owner$$

اجتماع دو رابطه نیز گاهی در عمل مورد نیاز است. فرض کنید که یک ورودی جدید بصورت $aisling \mapsto muckross$ اضافه شود. این وضعیت بصورت زیر نشان داده می شود.

$$home\_owner' = home\_owner \cup aisling \mapsto muckross$$

حال فرض کنید می خواهیم اسامی تمام خانم هایی را که مالک خانه هستند، داشته باشیم. بنابراین باید رابطه $home\_owner$ را محدود به حالاتی کنیم که عنصر اول زوج مرتب های آن، خانم باشند. توجه کنید که داریم $female : \mathbb{P}\, Person$ و $female \subseteq \{aisling, rebecca\}$.

$$home\_owner = \{aisling \mapsto muckross, rebecca \mapsto nirvana, lawrence \mapsto nirvana\}$$

$$female \lhd home\_owner = \{aisling \mapsto muckross, rebecca \mapsto nirvana\}$$

$female \lhd home\_owner$ رابطه ای است که زیر مجموعه $home\_owner$ است و در این رابطه ، اولین عنصر هر زوج مرتبی، $female$ است. عملگر $\lhd$ محدود کننده دامنه عبارت [۲] است و ویژگی اصلی آن بصورت زیر بیان می شود:

$$x \mapsto y \in U \lhd R \Leftrightarrow (x \notin U \land x \mapsto y \in R)$$

که در آن $R : X \leftrightarrow Y$ و $U : \mathbb{P} X$. عملگر دیگری تحت عنوان عملگر ضدمحدودیت دامنه [۳] وجود دارد که ویژگی اصلی آن بصورت زیر توصیف می شود:

$$x \mapsto y \in U \ntriangleleft R \Leftrightarrow (x \notin U \land x \mapsto y \in R)$$

که در آن $R : X \leftrightarrow Y$ و $U : \mathbb{P} X$. همچنین عملگرهای محدودیت برد [۴] با نماد $\rhd$ و ضدمحدودیت برد [۱] با نماد $\ntriangleright$ در زبان Z مورد استفاده قرار می گیرند. این عملگرها تعاریفی مشابه عملگرهای دامنه دارند با این تفاوت که برای برد تابع $x \mapsto y$ محدودیت ایجاد می کند.

---

[۲] restriction domain termed
[۳] anti-restriction domain
[۴] restriction range
[۱] range anti-restriction

```
┌─ TempMap ──────────────────────────
│ CityList : ℙ City
│ temp : City ⇸ Z
├────────────
│ dom temp = CityList
└────────────────────────────────────
```

۳.۳ توابع

یک تابع، بیانگر وابستگی بین اشیا نوع X با اشیا نوع Y می باشد که در آن هر شی از نوع X، تنها به یک شی از نوع Y وابسته است. به بیان دیگر می توان گفت که یک تابع مجموعه ای از زوج مرتب هاست که در آنها عنصر اول هر زوج مرتب، حداکثر با یک عنصر رابطه دارد. درحقیقت تابع نوع خاصی از رابطه است که در آن هریک از عناصر مجموعه دامنهف تنها با یک عنصر از مجموعه برد، می توانند رابطه داشته باشند. تابع ممکن است کامل یا جزئی باشد.
یک تابع جزئی از X به Y که به صورت $f : X \nrightarrow Y$ نشان داده می شود، تابع $f : X' \longrightarrow Y$ برای یک زیرمجموعه سره X' از X است. اگر زیرمجموعه X' سره نباشد، یعنی اگر $X' = X$ ، تابع f را یک تابع کامل می گوییم. از توابع جزئی معمولا زمانی استفاده می شود که دامنه یک تابع مشخص نیست. تابع جزئی بصورت زیر تعریف می شود:

$$\forall x : X;\ y, z : Y . (x \longmapsto y \in f \land x \longmapsto z \in f \Rightarrow y = z)$$

وابستگی بین x و y بصورت $f(x) = y$ نشان داده می شود. این بدان مفهوم است که مقدار تابع جزئیf برای x، برابر y است. تابع کامل از X به Y، که با $f : X \to Y$ نشان داده می شود، یک تابع جزئی است که در آن هر عنصر مجموعه X به یک مقدار از مجموعه Y وابسته شده است.

$$f : X \to Y \Leftrightarrow f : X \nrightarrow Y \land \operatorname{dom} f = X$$

واضح است که هر تابع کامل، یک تابع جزئی است ولی هر تابع جزئی، یک تابع کامل نیست. عملگری که از تکرار در توصیف ها ناشی می شود، عبارت است از عملگر لغو [1] . به توصیف

[1] override

نگاشت دمایی که در ?? آمده است توجه کنید.یک مثال از نگاشت دما بصورت زیر می تواند باشد:

$$temp = \{Cork \mapsto ۱۷, Dublin \mapsto ۱۹, London \mapsto ۱۵\}$$

حال مساله بروزرسانی دما مطرح می گردد، زمانیکه مثلا می خواهیم دمای Cork را تغییر دهیم. برای مثال $(Cork \mapsto ۱۸)$. بنابراین یک نمودار دمای جدید با استفاده از نمودارهای قدیمی و عملگر لغو تابع خواهیم داشت که نتیجه آن

$$\{Cork \mapsto ۱۸, Dublin \mapsto ۱۹, London \mapsto ۱۵\}$$

است. این عملیات بصورت زیر نوشته می شود:

$$temp' = temp \oplus Cork \mapsto ۱۸$$

عملگر لغو تابع، دو تابعی را که از دارای یک نوع هستند باهم ترکیب کرده و تابع جدیدی با همان نوع ایجاد می کند. تاثیر عملگر لغو به این صورت است که ورودی $\{Cork \rightarrow ۱۷\}$ از نمودار دما را حذف می کند و با ورودی $\{Cork \rightarrow ۱۸\}$ جایگزین می کند.
فرض کنید $f, g : X \nrightarrow Y$ دو تابع جزئی هستند. $f \oplus g$ اینگونه تعریف می شود که f بوسیله g لغو شده است. تعریف $f \oplus g$ بصورت زیر است:

$$(f \oplus g)(x) = g(x) where x \in \operatorname{dom} g$$
$$(f \oplus g)(x) = f(x) where x \notin \operatorname{dom} g \land x \in \operatorname{dom} f$$

این عملگر همچنین ممکن است بصورت زیر تعریف شود:

$$f \oplus g = ((\operatorname{dom} g) \lhd f) \cup g$$

در Z امضاهایی برای توابع injective، bijective و surjective وجود دارد. تابع injective یک تابع یک به یک است.

$$f(x) = f(y) \Rightarrow x = y$$

تابع surjective بصورت زیر تعریف می شود:

$$Given y \in Y, \exists x \in X such that f(x) = y$$

تابع bijective نیز یک تابع یک به یک است که نشان می دهد مجموعه های X و Y با یکدیگر تناظر یک به یک دارند.
Z شامل نشانگذاری عبارت لامبدا[۱]، برای تعریف توابع است. برای مثال تابع:

$$cube == \lambda x : N \bullet x * x * x$$

ترکیب توابع f و g مشابه ترکیب روابط است.

۴.۳ دنباله ها

نوه تمامی دنباله های عناصر از یک مجموعه مانند Xف با نماد $seqX$ نشان داده می شود. دنباله بصورت $< x_۱, x_۲, ..., x_n >$ نمایش داده می شود. دنباله تهی به صورت $<>$ نشان داده می شود. دنباله ها ممکن است برای نشان دادن تغیر حالت یک متغیر در طول زمان، بکار برده شوند که در آن هر عنصر دنباله، نشاندهنده مقداری از متغیر در زمان های گسسته است.
دنباله ها همان توابع هستند و یک دنباله از عناصر روی مجموعه X، یک تابع متناهی از مجموعه اعداد طبیعی به X است. یک تابع جزئی متناهی به نام f از X به Y بصورت $f : X \nrightarrow Y$ تعریف می شود.
یک دنباله متناهی از عناصر X بصورت $f : N \nrightarrow X$ نشان داده می شود. دامنه این تابع شامل تمام اعداد بین ۱ و $\#f$ است که در آن $\#f$ کاردینالیتی f است. این موضوع با استفاده از فرمول زیر نشان داده می شود.

$$seq\ \ X == \{f : N \nrightarrow X \mid \mathrm{dom}\ \ f = ۱..\#f \bullet f\}$$

دنباله $< x_۱, x_۲, ..., x_n >$ که در بالا معرفی شد بصورت $\{۱ \longmapsto x_۱, ۲ \longmapsto x_۲, ..., n \longmapsto x_n\}$ تعریف می شود.
عملگرهای متنوعی برای دستکاری کردن دنباله ها وجود دارند. یکی از این عملگرها، عملگر الحاق است. فرض کنید $\sigma =< x_۱, x_۲, ..., x_n >$ و $\tau =< y_۱, y_۲, ..., y_n >$ دو دنباله مفروض باشند. آنگاه

$$\sigma \cap \tau =< x_۱, x_۲, ..., x_n, y_۱, y_۲, ..., y_n >$$

---
[۱] Lambda

سرآیند [۲] یک دنباله غیرتهی، اولین عنصر آن دنباله است.

$$heads\ \ \sigma = head < x_1, x_2, ..., x_n >= x_1$$

ته[۳] یک دنباله غیرتهی، شامل تمامی عناصر دنبالهف غیر از عنصر اول آن، می باشد.

$$tail\ \ \sigma = tail < x_1, x_2, ..., x_n >=< x_2, x_3, ..., x_n >$$

فرض کنید $f : X \longrightarrow Y$ و یک دنباله بصورت $\sigma : seqX$ وجود دارد. آنگاه تابع نگاشت [۴] f را برای تمام عناصر σ بکار می برد:

$$map\ \ f\ \sigma = mapf < x_1, x_2, ..., x_n >=< f(x_1), f(x_2), ...f(x_n) >$$

تابع نگاشت، همچنین با استفاده از ترکیب توابع بصورت زیر تعریف می شود:

$$map\ \ f\ \sigma = \sigma;\ f$$

معکوس[۵] یک دنباله با استفاده از تابع rev بدست می آید:

$$rev\ \ \sigma = rev < x_1, x_2, ..., x_n >=< x_n, ..., x_2, x_1 >$$

۵.۳ کیسه ها

کیسه[۱] مشابه مجموعه است با این تفاوت که کیسه می تواند عنصر تکراری داشته باشد. یک کیسه از عناصر نوع X بصورت یک تابع جزئی تعریف می شود که دامنه آن، از نوع عناصر موجود در کیسه و برد آن، تمام اعداد مثبت است. تعریف یک کیسه از نوع X بصورت زیر است:

$$bag\ \ X = X \nrightarrow \mathbb{N}_1.$$

به عنوان مثال یک کیسه از مهره ها را در نظر بگیرید. این کیسه ممکن است شامل ۳ مهره آبی، ۲ مهره قرمز و یک مهره سبز باشد. این کیسه را می توان بصورت $B = [b, b, b, g, r, r]$ نشان داد. این کیسه از مهره ها همچنین بصورت زیر تعریف می شود:

$$bag\ \ Marbel == Marbel \nrightarrow \mathbb{N}_1.$$

---

hesd[۲] tail[۳] map[۴] reverse[۵] bag[۱]

$$\begin{array}{|l}
\hline \triangle Borrow \\
stock : bag\ \ Good \\
price : Good \rightarrow \mathbb{N}_1 \\
\hline
\mathrm{dom}\ \ stock \subseteq \mathrm{dom}\ \ price \\
\hline
\end{array}$$

شکل ۱.۳ توصیف دستگاه فروش با استفاده از کیسه

تابع $count$ تعداد رخدادهای یک عنصر موجود در کیسه را مشخص می کند. به عنوان مثال، در کیسه فوق، $count\ \ Marbel\ \ b = 3$ و $count\ \ Marbel\ \ y = 0$ ،بنابراین هیچ مهره زردی در کیسه وجود ندارد. این مطلب با استفاده از فرمول های زیر بیان می شود:

$$\begin{array}{ll}
count\ \ bagX\ \ y = 0 & y \notin bag\ \ X \\
count\ \ bagX\ \ y = (bagX)(y) & y \in bag\ \ X
\end{array}$$

عضو y در کیسه X قرار دارد اگر و فقط اگر y در دامنه کیسه X باشد:

$$y\ \ in\ \ bagX \Leftrightarrow y \in \mathrm{dom}(bagX)$$

اجتماع دو کیسه $B_1 = [b,b,b,g,r,r]$ و $B_2 = [b,g,r,y]$ بصورت $B_1 \uplus B_2 = [b,b,b,b,g,g,r,r,r,y]$ نوشته می شود. عمل اجتماع با استفاده از فرمول های زیر توصیف می شود:

$$\begin{array}{ll}
(B_1 \uplus B_2)(y) = B_2(y) & y \notin \mathrm{dom}\ \ B_1 \wedge y \in \mathrm{dom}\ \ B_2 \\
(B_1 \uplus B_2)(y) = B_1(y) & y \in \mathrm{dom}\ \ B_1 \wedge y \notin \mathrm{dom}\ \ B_2 \\
(B_1 \uplus B_2)(y) = B_1(y) + B_2(y) & y \in \mathrm{dom}\ \ B_1 \wedge y \in \mathrm{dom}\ \ B_2
\end{array}$$

کیسه ممکن است برای ضبط تعدا موجودی هر محصول در یک انبار که بخشی از یک سیستم فروش است، بکار برده شود. شمای فوق (شکل۱.۳)، تعداد اقلام باقیمانده از هر محصول را در یک سیستم فروش، مدلسازی می کند.

عملیات یک ماشین فروش نیازمند عملگرهایی نظیر عملگر شناسایی مجموعه سکه های قابل قبول، بررسی کافی بودن مبلغ ورودی متناسب با قیمت کالا، بازگرداندن مبلغ اضافی به مشتری، و بروزرسانی مقدار موجود از هر کالا پس از انجام عملیات خرید، می باشد.

۶.۳ شِما و ترکیب شِما

توصیف Z در یکسری جعبه های بصری ارائه می شود که آنها را شِما یا سکیما[۲] گویند. این جعبه ها در حالت های خاصی به کار می روند . نشان گذاری هایی را برای نمایش حالت قبل و حالت بعد، بکار می گیرند. (به عنوان مثال s و s' که در آن s' حالت بعد از s است). شِماها، تمامی اطلاعات مرتبط باهم را برای شرح یک حالت، گروهبندی می کنند.
عمگرهای مفیدی برای کار با شِماها وجود دارند مانند عملگر شمول شما[۳]، عملگر ترکیب شِما [۱] و استفاده از اتصال گزاره ها برای متصل کردن شِماها به یکدیگر.
نماد Δ بصورت قراردادی نشاندهنده این است که شِمای حاضر بر حالت تاثیر می گذارد. در مقابل عملگر Ξ به این معناست که حالت از شِما تاثیر نمی پذیرد. این قراردادها، قابلیت خوانایی توصیف را افزایش می دهند و امکان تعریف عملگرهای پیچیده تر را فراهم می کنند.
عملگر ترکیب شِما باعث می شود که شِماهی جدیدی از شِماهای موجود مشتق گردد. شِمایی با نام $S_۱$ ممکن است در بخش اعلان شِمای $S_۲$ مورد استفاده قرار گیرد. تاثیر شمول به این نحو است که اعلان های موجود در $S_۱$ ، حالا بخشی از اعلان های $S_۲$ هستند و گزاره های $S_۱$ و $S_۲$ با یکدیگر ترکیب عطفی شده اند. اگر یک متغییر هم در $S_۱$ و هم در $S_۲$ تعریف شده باشد، باید نوع آن در هر دو شما یکسان باشد.

$S_۱$
$x, y : \mathbb{N}$
$x + y > ۲$

$S_۲$
$S_۱;\ z : \mathbb{N}$
$z = x + y$

نتیجه اینکه $S_۲$ شامل اعلان ها و گزاره های $S_۱$ است (تصویر ۲.۳).

schema[۲] *schema inclusion*[۳] composition schema[۱]

$$
\begin{array}{|l}
\multicolumn{1}{l}{S_۲} \\
\hline
x, y : \mathbb{N} \\
z : \mathbb{N} \\
\hline
x + y > ۲ \\
z = x + y \\
\hline
\end{array}
$$

شکل ۲.۳ شمول دو شِما

$$
\begin{array}{|l}
\multicolumn{1}{l}{S} \\
\hline
x, y : \mathbb{N} \\
z : \mathbb{N} \\
\hline
x + y > ۲ \wedge z = x + y \\
\hline
\end{array}
$$

شکل ۳.۳ ترکیب فصلی دو شمای $S_۱$ و $S_۲$

دو شِما ممکن است توسط اتصال های گزاره ای نظیر $S_۱ \wedge S_۲$ ، $S_۱ \wedge S_۲$ ، $S_۱ \Rightarrow S_۲$ و یا $S_۱ \Leftrightarrow S_۲$ به یکدیگر متصل شوند. شِمای $S_۱ \wedge S_۲$ باعث می شود که بخش اعلان دو شِمای $S_۱$ و $S_۲$ با یکدیگر ادغام شوند و سپس گزاره های آنها نیز بوسیله عملگر منطقی ترکیب فصلی $\wedge$ با یکدیگرف ترکیب شوند. برای مثال $S = S_۱ \wedge S_۲$ در شکل ۳.۳ نشان داده شده است.

دو عملگر شمول شِما و اتصال شِماها، برای تبدیل زیر نوع ها به انواع بیشینه، از نرمال سازی استفاده می کنند. از طرفی گزاره ها نیز برای محدود کردن انواع بیشینه به زیرنوع ها بکار می روند. این عمل منجر به جایگزینی اعلان متغیرها می شود. به عنوان مثال $u : ۱..۳۵$ با $u : \mathbb{Z}$ جایگزین می شود و گزاره $u > ۰ \;\; and \;\; u < ۳۶$ به بخش گزاره شِما اضافه می گردد.

دو نماد Δ و Ξ به کرات در تعریف شِماها بکار می روند. وقتی که نشان گذاری $\Delta TempMap$ در توصیف یک شِما، دیده می شود به این مفهوم است که این شِما، حالت را تغییر می دهد.

$$\Delta TempMap = TempMap \wedge TempMap'$$

شکل مفصل تر $\Delta TempMap$ در تصویر۴.۳ توصیف شده است.

$$
\begin{array}{|l}
\multicolumn{1}{l}{\Delta TempMap} \\
\hline
CityList, CityList' : \mathbb{P}\ City \\
temp, temp' : City \nrightarrow Z \\
\hline
\operatorname{dom} temp = CityList \\
\operatorname{dom} temp' = CityList' \\
\hline
\end{array}
$$

شکل ۴.۳ کاربرد عملگر Δ

$$
\begin{array}{|l}
\multicolumn{1}{l}{\Xi TempMap} \\
\hline
\Delta TempMap \\
\hline
CityList = CityList' \\
temp = temp' \\
\hline
\end{array}
$$

شکل ۵.۳ کاربرد عملگر Ξ

جدول ۱.۳ مراحل ترکیب شِما

گام	روال
۱.	تمامی مقادیر حالت "بعدی" موجود در S، با نام های جدید، بازنامگذاری می شوند: $S[s^+/s']$
۲.	تمامی مقادیر حالت "قبلی" موجود در T با همان نام های جدید، بازنامگذاری می شوند: $S[s^+/s]$
۳.	ترکیب عطفی دو شِمای جدید ایجاد می شود: $S[s^+/s'] \wedge T[s^+/s]$
۴.	متغیرهایی که در گام های ۱ و ۲ معرفی شده اند، مخفی می شوند. $S;\ T = (S[s^+/s'] \wedge T[s^+/s])(s^+)$

نشان گذاری Ξ برای توصیف عملگرهایی استفاده می شود که منجر به تغییر حالت نمی شوند. نمونه ای از این نشان گذاری در شکل ۵.۳ قابل مشاهده است.

عملگر ترکیب شِما باعث می شود که توصیف های جدیدی از روی توصیف های موجود ساخته شوند. ترکیب شِما، مقادیر حالت بعدی یک شِما را به مقادیر حالت قبلی شِمای دیگری، ربط می دهد. ترکیب دو شِمای S و T ، $(S;\ T)$ شامل چهار مرحله است که در جدول ۱.۳ نشان داده شده است.

چهار شمای S، T، $T_۱$ و $S_۱$ در شکل ۶.۳ نشان داده شده است.

این شِماها در تصویر ۷.۳ با یکدیگر ترکیب شده اند.

$S_۱$ و $T_۱$ نتایج گام های ۱ و ۲ را نشان می دهند. x' در S با s^+ بازنامگذاری شده است و x در T با x^+ بازنامگذاری شده است. نتایج گام های ۳ و ۴ را نیز در تصویر ۷.۳ می توانید مشاهده کنید.

S

$x, x', y? : \mathbb{N}$

$x' = y? - ۲$

T

$x, x' : \mathbb{N}$

$x' = x + ۱$

$S_۱$

$x, x^{+}, y? : \mathbb{N}$

$x^{+} = y? - ۲$

$T_۱$

$x^{+}, x' : \mathbb{N}$

$x' = x^{+}۱$

شکل ۶.۳ معرفی چهار شِمای S، T، $S_۱$ و $T_۱$

$S_1 \wedge T_1$

$$x, x^{+}, x', y? : \mathbb{N}$$

$$x^{+} = y? - 2$$
$$x' = x^{+} 1$$

$S;\ T$

$$x, x', y? : \mathbb{N}$$

$$\exists x^{+} : \mathbb{N} \bullet$$
$$(x^{+} = y? - 2$$
$$x' = x^{+} + 1)$$

شکل ۷.۳ ترکیب شِما

۴
چند نمونه ساده توصیف Z

۱.۴ نمونه اول: کتابچه تولد

بهترین راه برای درک زبان Z مطالعه نمونه های ساده است. به عنوان اولین نمونه، سیستمی پیاده سازی می شود که در آن کتابچه تولد، به جای استفاده از دفترچه و خودکار، توسط یک سیستم کامپیوتری ایجاد می شود. در این سیستم، تاریخ تولد افراد ثبت می شود و سیستم قادر است که با نزدیک شدن روز تولد افراد، آن را یادآوری کند.
شخصی که می خواهد برای خود در سیستم کتابچه تولد، یک حساب کاربری ایجاد کند، باید نام افراد و تاریخ تولد آنها را ثبت کند. بنابراین مجموعه ای از نام ها و مجموعه ای از تاریخ ها را، به عنوان نوع اصلی، در این توصیف خواهیم داشت.

$$[NAME, DATE]$$

تعریف این دو مجموعه باعث می شود که بتوان مجموعه ها را بدون بیان صریح نوع اشیایی که

$$
\begin{array}{|l}
\hline
\multicolumn{1}{l}{BirthdayBook} \\
known : \mathbb{P}\ NAME \\
birthday : NAME \pfun DATE \\
\hline
known = \mathrm{dom}\ \ birthday \\
\hline
\end{array}
$$

شکل ۱.۴

شامل می شوند، نامگذاری کرد.

اولین جنبه سیستم، تشریح فضای حالت آن است که با استفاده از شمای شکل ۱.۴ توصیف شده است.

مشابه دیگر شِماها، این شِما نیز شامل دو بخش است که با یک خط تقسیم مرکزی از یکدیگر جدا شده اند. در بخش بالا، متغیرها اعلان شده اند و در بخش پایین، رابطه بین متغیرها و مقادیرشان مشخص شده است. در این مورد، فضای حالت سیستم و دو متغیری که نشاندهنده مشاهدات بااهمیت هستند و می توانند حالت ها را ایجاد کنند، تشریح شده اند:

: *known*• مجموعه ای از نام هاست که تاریخ تولد آنها ثبت شده است.

: *birthday*• تابعی است که زمانیکه نام معینی مشخص می شود، تاریخ تولد منتسب به آن را می دهد.

بخشی از شِما که در زیر خط قرار دارد، رابطه ای را نشان می دهد که در تمامی حالت های سیستم، درست است و با پس از اعمال عملگرها نیز همچنان این رابطه درست باقی می ماند. در این مثال، رابطه بیانگر این است که مجموعه *known* مشابه دامنه تابع *birthday* است. این متغیر شامل مجموعه ای از نامهاست که برای آنها تاریخ تولدی ثبت شده است. این رابطه، در سیستم غیرقابل تغییر است.

در این مثال، غیرقابل تغییر بودن، به مقدار متغیر *known* اجازه می دهد که از مقدار *birthday* مشتق شود. در واقع *known* یک مولفه مشتق شده از حالت است و می توان سیستم را بدون اشاره به *known* مشخص کرد. البته باید توجه داشت که دادن نام ها، خوانایی

توصیف را افزایش می دهد زیرا یک دید انتزاعی از فضای حالت کتابچه تولد را ایجاد می کند. یکی از حالت های ممکن سیستم حالتی است که شامل سه نفر در مجموعه $known$ است که تاریخ های تولد آنها با استفاده از تابع *birthday* ثبت شده است:

$$known = \{John, Mike, Susan\}$$

$$\begin{aligned} birthday = \{&John \mapsto ۲۵ - Mar, \\ &Mike \mapsto ۲۰ - Dec, \\ &Susan \mapsto ۲۰ - Dec\}. \end{aligned}$$

ویژگی غیرقابل تغییر بودن در مثال فوق ارضا می شود زیرا تابع *birthday* دقیقا برای هر سه اسم نام موجود در *known*، یک روز را ثبت کرده است.
توجه کنید که در شرح فضای حالت سیستم، محدودیتی برای تعداد زمان تاریخ تولدهای ثبت شده در کتابچه تولد، قرار داده نشده است. همچنین فرمت خاصی برای ورود نام ها و روزهای تولد تعریف نشده است. از طرف دیگر برای هر فرد تنها یک تاریخ تولد ثبت می شود زیرا که *birthday* یک تابع است، اما برای دو نفر متفاوت ممکن است یک تاریخ تولد ثبت شود (همانشور که در مثال فوق نیز این اتفاق رخ داده است).
در ادامه عملگرهای سیستم تعریف خواهند شد. اولین عملگر، عملگری است که امکان اضافه کردن یک تاریخ تولد جدید را فراهم می کند. این عملگر در شِمای شکل۲.۴ توصیف شده است.

اعلان $\Delta Birthday$ بیانگر این مطلب است که این شِما، یک تغییر حالت را توصیف می کند. در این شِما چهار متغیر معرفی می شوند: $known$، $birthday$، $known'$ و $birthday'$. دوتای اول مربوط به مشاهدات حالت، قبل از تغییر و دوتای بعدی، مربوط به مشاهدات حالت، بعد از تغییر هستند. هر زوج متغیر، محدودیت غیرقابل تغییر بودن را ارضا می کنند. سپس دو ورودی عملگر، اعلان می شوند. برای اعلان ورودی ها، پس از اعلان نام ورودی، علامت سوال گذاشته می شود.
بخشی از شِما که در زیر خط قرار گرفته است، ابتدا به بیان یک پیش شرط برای انجام موفقیت

$$\begin{array}{|l}
\multicolumn{1}{l}{__ AddBirthday ________________________________} \\
\Delta BirthdayBook \\
name? : NAME \\
date? : DATE \\
\hline
name? \notin known \\
birthday' = birthday \cup \{name? \mapsto date?\} \\
\hline
\end{array}$$

شکل ۲.۴

آمیز عملگر می پردازد، به این ترتیب که نامی که می خواهد اضافه شود نباید قبلا در سیستم ثبت شده باشد. این پیش شرط منطقی به نظر می رسد زیرا به ازای هر نفر تنها یک تاریخ تولد باید در سیستم ثبت شود. اینکه اگر پیش شرط برآورده نشود، چه اتفاقی خواهد افتاد، در این توصیف مشخص نشده است.
اگر پیش شرط برآورده شود در خط بعدی تابع $birthday$ توسعه داده می شود و یک نام جدید به تاریخ تولدش نگاشت می شود. انتظار می رود که نام جدید نیز به مجموعه نام های موجود در سیستم اضافه شود.

$$known' = known \cup \{name?\}.$$

دروقع این مطلب از توصیف $AddBirthday$ ، با استفاده از ویژگی غیرقابل تغییر بودن حالت، پیش و پس از اعمال عملگر، قابل اثبات است.

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

ویژگی های اثبات و حالت، مانند آنچه که در بالا ذکر شد، تضمین میکنند که توصیف ها از دقت بالایی برخوردارنند که در نهایت منجر به توسعه سیستم هایی می گردد که درنهایت رفتاری بدون اشکال خواهند داشت.

$$
\begin{array}{|l}
\multicolumn{1}{l}{FindBirthday} \\
\hline
\Xi BirthdayBook \\
name? : NAME \\
date! : DATE \\
\hline
name? \in known \\
date! = birthday(name?) \\
\hline
\end{array}
$$

شکل ۳.۴

دو واقعیت درخصوص dom در این استدلال وجود دارد که مثالی از یک قانون ریاضی است. این قانون عبارت است از:

$$
\begin{aligned}
&dom(f \cup g) = (dom\ \ f) \cup (dom\ \ g) \\
&dom\{a \mapsto b\} = \{a\}.
\end{aligned}
$$

عملگر دیگری که برای یافتن روز تولد افراد بکار می رود، با استفاده از شِما شکل ۳.۴ تعریف شده است. در شِمای شکل ۳.۴ دو علامت جدید دیده می شود. اعلان $\Xi BirthdayBook$ بیانگر این است که این عملگر منجر به تغییر حالت نخواهد شد. به این معنا که مقدار $known'$ و $birthday'$ ، در مشاهدات بعد از اعمال عملگر، برابر است با مقدار $kKnown$ و $birthday$ قبل از اعمال عملگر. استفاده از $\Xi BirthdayBook$ در خط اول شِما تاثیری مشابه این دارد که در خط اول $\Delta BirthdayBook$ آورده شود و دو تساوی زیر نیز در ادامه و در زیر آن نوشته شوند:

$$
\begin{aligned}
&known' = known \\
&birthday' = birthday
\end{aligned}
$$

علامت جدید دیگری که در این شِما استفاده شده است، نمادی است که بعد از $date$ آمده است و تعیین می کند که این متغیر یک خروجی است. عملگر $FindBirthday$ نام را به عنوان ورودی دریافت م کند و تاریخ تولد متناسب با آن را به عنوان خروجی، می دهد. پیش شرطی که برای انجام موفقیت آمیز این عملگر مورد نیاز است این است که $name?$ یکی از نام های

$$\begin{array}{|l}
\multicolumn{1}{l}{\underline{\quad Remind \quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad}} \\
\Delta BirthdayBook \\
today? : DATE \\
cards! : \mathbb{P}\ NAME \\
\hline
cards! = \{n : \ known \mid \ birthday(n) = \ today?\} \\
\hline
\end{array}$$

شکل ۴.۴

موجود در سیستم باشد. در اینصورت خروجی $date!$ برابر است با مقدار تابع $birthday$ که برای آرگومان $name?$ در نظر گرفته شده است.

یکی از پرکاربردترین عملگرهای موجود در این سیستم، عملگری است که جهت یادآوری روز تولد استفاده می شود. این عملگر هر روز، اسامی افرادی را که در آن روز متولد شده اند، اعلام می کند. این عملگر یک ورودی تحت عنوان $today?$ و یک خروجی تحت عنوان $cards!$ دارد که مجموعه ای از نام هاست که می تواند صفر، یک، دو و یا تعداد بیشتری از افرادی باشد که روز تولدشان برابر با روز تعیین شده است و باید در این روز برای آنها کارت تولد فرستاد. در شِمای شکل ۴.۴ که به توصیف عملگر $Remind$ می پردازد نیز از علامت Δ استفاده شده است که بیانگر این است که حالت تغییر نمی کند. این شِما پیش شرط ندارد. خروجی $cards!$ برابر است با مجموعه ای از تمام مقادیر n از مجموعه $known$ که مقدار تابع $birthday$ آنها برابر با $today?$ است. به بیان کلی تر، y عضوی از مجموعه $\{x : S \mid ...x...\}$ است اگر y عضوی از S باشد و شرط $...y...$، با جایگذاری y به جای x بدست آید. در واقع می توان گفت:

$$y \in \{x : S \mid ...x...\} \Leftrightarrow y \in S \wedge (...y...).$$

که در مثال ما:

$$\begin{array}{l} m \in \{n : known \mid birthday(n) = today?\} \\ \quad\quad \Leftrightarrow m \in known \wedge birthday(m) = today?. \end{array}$$

یک نام m در مجموعه خروجی $cards!$ قرار دارد اگر و فقط اگر در سیستم وجود داشته باشد و روز تولدی که برای آن ثبت شده است برابر با $today?$ باشد.

$$
\begin{array}{|l}
\multicolumn{1}{l}{\_\_ InitBirthdayBook \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_} \\
BirthdayBook \\
\hline
known = \varnothing \\
\hline
\end{array}
$$

شکل ۵.۴

برای به پایان رساندن این توصیف، باید حالت شروع سیستم را بیان کرد. این حالت، حالت اولیه [۱] سیستم می گویند و بوسیله شِمای شکل ۵.۴ توصیف می شود. در این شِما، کتابچه تولد به نحوی توصیف میشود که در آن مجموعه $known$ خالی درنظر گرفته شده است. همچنین تابع $birthday$ نیز تهی درنظر گرفته می شود.

یک پیاده سازی صحیح توصیفی که روزهای تولد را ذخیره می کند و نمایش می دهد، باید بنحوی باشد که هیچگونه ورودی اشتباهی به آن وارد نشود. اما این توصیف یک ایراد جدی دارد: زمانیکه کاربر سعی می کند تاریخ تولد شخصی را به سیستم وارد کند که در حال حاضر در سیستم وجود دارد، یا سعی در تاریخ تولد شخصی داشته باشد که در سیستم وجود ندارد. برای این وضعیت ها، هیچ حالتی در سیستم توصیف نشده است. در این وضعیت ها سیستم باید یک رفتار مسئولیت پذیر از خود بروز دهد به این نحو که ورودی های غلط را نادیده بگیرد.

در توصیف سیستم کتابچه تاریخ تولد، رفتار سیستم در قبال ورودی های صحیح، بطور واضح و شفافی بیان شده است. این توصیف نیازند تغییراتی که بتواند ورودی های غلط را نیز مدیریت کند. در واقع سیستم باید موقعیت های خطا را شناسایی کند و در صورت بروز هریک از آنها، رفتاری متناسب از خود بروز دهد. با اصلاح توصیف ها، به توصیف های قوی تری خواهیم رسید که عملیات حساب شِمای Z را بکار می برند.

خروجی با عنوان $result!$ را به تمامی عملیات های عریف شده برای سیستم، اضافه می کنیم. اگر عملیات با موفقیت به پایان رسید، این خروجی ارزش ok خواهد گرفت، اما اگر خطایی شناسایی شد، دو مقدار $already\_known$ و not_known به آن تخصیص داده می شود. تعریف نوع آزاد [۱] برای $REPORT$ که دقیقا شامل سه مقدار می باشد، در ادامه آمده است.

[1]initial state [1]free type

$$
\begin{array}{|l}
\multicolumn{1}{l}{\underline{\ \ Success\ }} \\
result! : \ \ REPORT \\
\hline
result! = ok \\
\hline
\end{array}
$$

شکل ۶.۴

$$REPORT ::= ok \mid already\_known \mid not\_known.$$

در شکل ۶.۴ شِمای $Success$ تعریف شده است که به توصیف وضعیتی می پردازد که در آن نتیجه ok است. عملگر ترکیب عطفی در حساب شِما، این اجازه را می دهد که دو شمای $AddBirthday$ و $Success$ به صورت زیر با یکدیگر ترکیب شوند.

$$AddBirthday \land Success$$

ترکیب این دو شِما، به شرح عملیاتی می پردازد که در آن برای ورودی صحیح، دو عملی که بوسیله $AddBirthday$ و $Success$ توصیف شده اند، رخ دهد و نتیجه ok شود. برای هر خطایی که ممکن است در ورودی رخ دهد، باید شِمایی تعریف کرد که گزارشی مناسب را در صورت رخداد خطا ارائه دهد. شِمای مربوط به گزارش *already_known* در شکل ۷.۴ نشان داده شده است. این خطا زمانی رخ می دهد که ورودی $name?$ عضوی از مجموعه $known$ باشد.

اعلان $\Xi BirthdayBook$ بیان می کند که در صورت بروز خطا، حالت سیستم تغییر نمی کند. می توان این توصیف را با توصیف هایی که پیش تر بیان شدند، ترکیب کرده و نسخه کامل تری از $AddBirthday$ را بدست آورد.

$$RAddBirthday \cong (AddBirthday \land Success) \lor AlreadyKnown.$$

این تعریف شِمای جدیدی را معرفی می کند که $RAddBirthday$ نام دارد. این شِما با ترکیب سه شِما در سمت راست رابطه، بدست آمده است. اگر ورودی $name?$ در حال حاضر وجود

$$
\begin{array}{|l}
\hline
\textit{AlreadyKnown} \\
\Xi BirthdayBook \\
name? : NAME \\
result! : REPORT \\
\hline
name? \in Known \\
result! = already\_known \\
\hline
\end{array}
$$

شکل ۷.۴

داشته باشد، حالت سیستم تغییر نمی کند و نتیجه $already\_known$ برگردانده می شود. در غیر اینصورت، تاریخ تولد جدید، با استفاده از $AddBirthday$ به پایگاه داده اضافه می شود و و نتیجه ok برگردانده می شود.

برای انجام عملیات $RAddBirthday$ باید نیازهای متنوعی را توصیف کرد و سپس این نیازها را در یک توصیف با یکدیگر ترکیب کرد. این ترکیب در نهایت رفتار عملیات را نشان می دهد. این بدان معنا نیست که نیازها بصورت جداگانه پیاده سازی شوند و پیاده سازی های مختلف با یکدیگر ترکیب گردند. در واقع این پیاده سازی سعی می کند که مکان مناسبی را برای تاریخ تولد جدید پیدا کند و همزمان هم بررسی می کند که نام ورودی پیش تر در پایگاه داده ثبت نشده باشد. کد مربوط به عملیات اضافه کردن ورودی جدید و بررسی خطا، باید با یکدیگر ترکیب شوند.

عملیات $RAddBirthday$ بطور مستقیم و با نوشتن یک شِما، توصیف می شود. این شِما از ترکیب گزاره های سه شِمای $AddBirthday$ ، $Success$ و $AlreadyKnown$ بدست می آید. تاثیر عملگر $\vee$ در ساخت شِما این است که گزاره های شِمای جدید از ترکیب فصلی گزاره های دو شِمای دیگر بدست می آید. بطور مشابه، تاثیر عملگر $\wedge$ نیز منجر به ترکیب عطفی دو گزاره خواهد شد. متغیرهای دو شِما نیز با یکدیگر ادغام می شوند. در این مثال، ورودی $name?$ و خروجی $result!$ و چهار مشاهده مختلف از حالت های قبل و بعد از اعمال عملگر، با استفاده از دو آرگومان $\vee$ ، به اشتراک گذاشته می شوند. با توجه به اینکه

$$\begin{array}{|l}
\multicolumn{1}{l}{RAddBirthday} \\
\hline
\Delta BirthdayBook \\
name? : NAME \\
date? : DATE \\
result! : REPORT \\
\hline
(name? \notin Known \land \\
birthday' = birthday \cup \{name? \mapsto date?\} \land \\
result! = ok) \\
(name? \in known \land \\
birthday' = birthday \land \\
result! = already_known) \\
\hline
\end{array}$$

شکل ۸.۴

برای توصیف $RAddBirthday$ از یک تک شِما استفاده شد، لازم است که بطور صریح بیان شود که حالت سیستم در وضعیت بروز خطا، تغییر نخواهد کرد. این مساله پیش تر با اعلان $\Xi BirthdayBook$ ، بطور ضمنی، بیان می شد.

نسخه تکامل یافته عملیات $FindBirthday$ باید بتواند درصورتیکه نام مورد جستجو در پایگاه داده وجود نداشته باشد، این مورد را گزارش دهد. این گزارش خطا در شِمای شکل ۹.۴ نشان داده شده است.

عملیات تکامل یافته $FindBirthday$ نیز که بصورت شِمای $RFindBirthday$ تعریف می شود، با استفاده از $FindBirthday$ و گزارش های $Success$ و $NotKnown$ قابل توصیف است.

$$RFindBirthday \cong (FindBirthday \land Success) \lor NotKnown.$$

عملیات $Remind$ در هر زمانی می تواند فراخوانده شود. این عملیات هیچگاه منجر به خطا نمی شود اما نسخه تکامل یافته آن نیازمند اضافه کردن گزارش موفقیت است.

$$RRmind \cong Remind \land Success.$$

$$
\begin{array}{|l}
\textit{NotKnown} \\
\hline
\Xi BirthdayBook \\
name? : NAME \\
result! : REPORT \\
\hline
name? \notin Known \\
result! = not\_known \\
\hline
\end{array}
$$

شکل ۹.۴

$$
\begin{array}{|l}
[CHAR] \\
\hline
blank : \mathbb{P} CHAR \\
\hline
TEXT == seq\ CHAR \\
SPACE == seq_1\ blank \\
WORD == seq_1\ (CHAR \setminus blank) \\
\hline
\end{array}
$$

شکل ۱۰.۴

۲.۴ نمونه دوم: تجزیه متن به واژگان

یک متن شامل دنباله ای از کاراکترهاست. یکی از کاراکترهای خاص که در متن بسیار استفاده می شود، کاراکتر فضای خالی [2] است. کاراکترهای فاصله [3] ، شکست خط [1] و tab نیز نوع خاصی از فضای خالی هستند. درحقیقت، واژه، دنباله ای از کاراکترها، غیر از فضای خالی، است. بنابرای فضای خالی کاراکتری است که دو واژه را از هم جدا می کند. فاصله، دنباله ای از کاراکترهای فضای خالی است. TEXT ممکن است شامل دنباله ای تهی از کاراکترها باشد.SPACE و WORD باید شامل حداقل یک کاراکتر باشند. به همین دلیل آنها را بصورت seq_1 اعلان کردیم.

تابع شمارش کلمات که با نام *words* نامگذاری شده است. تابع *words* دنباله ای از تمام واژه های موجود در متن را بازمی گرداند. به عنوان مثال :

[2] Blank [3] space [1] line break

$$
\begin{array}{|l}
words : TEXT \rightarrow seq \;\; WORD \\
\hline
\forall s : SPACE; \quad w : WORD; \quad l, r : TEXT; \\
words <> = <> \wedge \\
words \;\; s = <> \wedge \\
words \;\; w = < w > \wedge \\
words(s \frown r) = words \;\; r \wedge \\
words(l \frown s) = words \;\; l \wedge \\
words(l \frown s \frown r) = (words \;\; l) \frown (words \;\; r)
\end{array}
$$

شکل ۱۱.۴

$$words < H, o, w, , a, r, e, , y, o, u, ? > = << H, o, w >, < a, r, e >, < y, o, u >>$$

واضح است که تابع$words$ تابعی از $TEXT$ به دنباله ای از $WORD$ است. برای تعریف تابع $words$ تمام الگوهای ممکن واژگان و فاصله ها در نظر گرفته می شود و برای هرکدام، یک تساوی نوشته می شود. همانطور که در شکل ۱۱.۴ می بینید، الگوهای زیادی وجود ندارد. زمانیکه متن خالی باشد، نتیجه نیز خالی است. زمانیکه متن تنها شامل کاراکتر فاصله باشد، باز هم نتیجه خالی است. زمانیکه متن شامل تنها یک کلمه است، نتیجه دنباله ای است که شامل تنها یک کلمه می باشد. زمانیکه متن تنها شامل یک کلمه به همراه یک فاصله در ابتدا یا انتهای آن باشد، نتیجه مشابه حالت قبل است (یعنی دنباله ای که شامل یک کلمه است). و درنهایت اینکه هرزمان که متن شامل یک کاراکتر فاصله باشد، آنرا نادیده گرفته و متن را از محل فاصله، به دو بخش می شکنیم.

این مثال تکنیک های مختلف موجود در Z را نشان می دهد که تعاریفی کوتاهتر و واضح تر از کد را ایجاد می کنند. تابع الگویی نظیر $l \frown s \frown r$ را بکار می برد که ساختار داخلی آرگومان هایشان را آشکار می کند.

تعداد کلمات موجود در متن t بصورت $\#(words \;\; t)$ نشان داده می شود. می توان تابعی تعریف کرد که مشابه کلمات، متن را به خطوط تشکیل دهنده اش بشکند. در چنین مثالی به جای فاصله، کاراکتر شکست خط را به عنوان یک کاراکتر خاص در نظر می گیریم و آن را nl

$$
\begin{array}{|l}
lines : TEXT \to seq\ \ LINE \\
\hline
...definition\ \ omitted...
\end{array}
$$

شکل ۱۲.۴

$$
\begin{array}{|l}
wc : TEXT \to (\mathbb{N} \times \mathbb{N} \times \mathbb{N}) \\
\hline
\forall file : TEXT \bullet \\
wc\ \ file = (\#(lines\ \ files),\ \ \#(words\ \ file),\ \ \#file) \\
wc == (\lambda\ \ file : TEXT \bullet (\#(lines\ \ file),\ \ \#(words\ \ file),\ \ \#file))
\end{array}
$$

شکل ۱۳.۴

می نامیم. چنین توصیفی را در شکل ۱۲.۴ می توان مشاهده کرد.

اکنون تمام آنچه را که برای توصیف رسمی عمل شمارش کلمات *Unix* نیاز داریم، در اختیار داریم. تابعی که برای این مورد طراحی می شود را *wc* می نامیم که آرگومان آن اسم یک فایل است و نتیجه آن یک چندتایی است که مولفه های آن، تعداد خطوط، کلمات و کاراکترهای موجود فایل می باشند. استفاده از این تابع بصورت زیر است.

```
% wc structure.tex
۱۱۰ ۵۵۹ ۴۵۰۹
```

در شکل ۱۹.۴ تعریف *wc* آمده است. تقریبا اکثر ویراستارهای متن، عملیات *Fill* را فراهم می کنند. عملیات *Fill* متنی را که در آن خطوط، طول های مختلفی دارند، به متنی تبدیل می کند که طول خطوط در آن تقریبا یکسان است. این عملیات موجب زیباتر شدن متن می گردد.

به عنوان مثال متن زیر را در نظر بگیرید:

operation fill The operation. fill a provides editor text any Almost
nicely into lengths different of lines with text raggedy-looking transforms

$$\begin{array}{|l}
\multicolumn{1}{l}{\underline{\mathit{Format}\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}}\\
\mathit{width} : \mathbb{N} \\
t, t' : \mathit{TEXT} \\
\hline
\mathit{words}\ \ t' = \mathit{words}\ \ t \forall l : \mathbf{ran}(\mathit{lines}\ \ t') \bullet \# l \leqslant \mathit{width} \\
\hline
\end{array}$$

شکل ۱۴.۴

length. same the nearly lines with text formatted

پس از اعمال عملیات *Fill* متن به صورت زیر تبدیل می شود:

Almost any text editor provides a fill operation. The fill operation transforms raggedy-looking text with lines of different lengths into nicely formatted text with lines nearly the same length.

اکنون به سراغ تعریف عملگر *Fill* می رویم. در واقع *Fill* تنها مثالی از عملیات *Format* است که ظاهر متن را تغییر می دهد. *Format* این کار را با شکستن خطوط به بخش های مختلف، گسترش یا انقباض فاصله بین کلمات، مشروط به اینکه هیچ خطی از عرض صفحه تجاوز نکند، انجام می دهد. توجه کنید که عملیات *Format* نباید محتوای متن را تغییر دهد. در واقع این عملیات همان کلمات را به ترتیب اصلی خود در متن اصلی، حفظ می کند (شکل ۱۴.۴).

عملیات *Fill* همان عملیات *Format* است که محدودیت اضافی را برآورده می کند. این محدودیت به این صورت است که خطوط تا حد امکان باید پر شوند. روش های مختلفی برای بیان این مطلب وجود دارد که هریک ظاهر متفاوتی به متن می دهند. شاید ساده ترین قانون این باشد که متن پر شده حداقل خطوط ممکن را اشغال کند. تعریف شکل ۱۵.۴ نشان می دهد که *Fill* یک عملیات کمینه سازی است. این یک نوع خاص از *Format* است که تعداد خطوط را به حداقل می رساند. درک این شِما کمی دشوار است زیرا به دو روش مختلف از *Format* استفاده می کند و رخدادهای مختلف t' نشاندهنده موارد مختلف هستند. t' در سمت چپ تساوی به معنای حالت پایانی *Fill* است. t' در داخل تعریف مجموعه، متغیری محدود است که

$$
\begin{array}{|l}
\multicolumn{1}{l}{\underline{\mathit{Fill}\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}}\\
\mathit{Format}\\
\hline
\#(\mathit{lines}\ \ t') = \mathit{min}\{t' : \mathit{TEXT} \mid \mathit{Format} \bullet \#(\mathit{lines}\ \ t')\}\\
\hline
\end{array}
$$

شکل ۱۵.۴

همه حالت های پایانی *Format* را که از حالت شروع *Fill* می توان به آنها رسید، گسترش می دهد. در داخل تعریف مجموعه، *Format* به عنوان یک گزاره بکار می رود. t در این *Format* حالت اولیه شِمای *Fill* است.

Fill غیرقطعی است. روش های مختلف زیادی برای شکستن خطوط و فاصله ها وجود دارند که تعداد خطوط را به حداقل می رسانند. در توصیف، غیرقطعیت معمولا چیز خوبی اسن. تعاریف غیرقطعی اغلب کوتاه تر و واضح تر هستند. این تعاریف جزئیات غیرضروری را حذف می کنند. زمانیکه به سراغ پیاده سازی می رویم، این تعاریف ما را در انتخاب، آزاد می گذارند و موجب افزایش بهره وری می شوند.

۳.۴ نمونه سوم: سیستم کنترل اسناد

در اینجا مدلی برای یک سیستم کنترل اسناد ساده در Z ارائه شده است. افرادی که با یکدیگر کار می کنند، نیازمند به اشتراک گذاری کارهایشان هستند اما این موضوع ممکن است منجر به ایجاد سوء تفاهم ها و سردرگمی هایی گردد. زمانیکه دو نفر بر روی یک چیز مشترک در حال کار هستند، ممکن است خطاهایی رخ دهد و انجام تغییرات منجر به ایجاد تصادم و تداخل با دیگری گردد (به عنوان مثال کار مشترک بر روی یک فایل برنامه نویسی). می توان از کامپیوتر برای جلوگیری کردن از بروز چنین خطاهایی استفاده کرد. در واقع این هدف یک سیستم کنترل اسناد است. دو مثال واقعی برای چنین سیستم هایی عبارتند از سیستم کنترل کد مبدا [۲] و سیستم کنترل بازبینی [۱] .

[2]Source Code Control System (SCCS) [1]Revision Control System (RCS)

در اینجا قوانین غیررسمی سیستم آورده شده است:
۱ـ اگر کاربری بخواهد یک سند را به ترتیب تغییراتی که در آن رخ داده، بررسی کند و این کاربر اجازه تغییر سند را نیز داشته باشد، و هیچ شخص دیگری در آن لحظه سند را تغییر ندهد، آنگاه آن کاربر می تواند سند را بررسی کند.
۲ـ به محض اینکه کاربری سندی را برای ویرایش بررسی کند، همه افراد دیگر از بررسی آن سند امتناع می کنند. البته افرادی که دارای مجوز خواندن هستند، می توانند سند را بخوانند.
۳ـ زمانیکه کاربر، سند را ویرایش کرد، باید آن را بررسی کند و به کاربرهای دیگر نیز امکان بررسی را بدهد.
در ادامه مدل Z مربوط به این سیستم آورده شده است. در ابتدا با معرفی دو مجموعه پایه شروع می کنیم. این دو مجموعه عبارتند از مجموعه افراد و مجموعه اسناد.

[PERSON، DOCUMENT]

برخی از افراد اجازه تغییر اسناد خاصی را دارند. می توان این ویژگی را به عنوان رابطه بین اسناد و افراد مدل کرد.

$permission : DOCUMENT \leftrightarrow PERSON$

این رابطه بصورت یک مجموعه از زوج مرتب هایی به شکل (سند، شخص) است. به عنوان مثال، فردی به نام مریم، می تواند توصیف را تغییر دهد، مریم و علی میتوانند طراحی را تغییر دهند و علی و احمد، امکان تغییر کد را دارند.

Maryam، Ali، Ahmad: PERSON
spec، design، code: DOCUMENT
permission = {(spec، Maryam)، (design، Maryam)، (design، Ali)، (code، Ali)، (code، Ahmad)}

وضعیت سیستم منجر به تعریف رابطه دیگری از نوع رابطه ای که در بالا ذکر شد، می شود. این رابطه به این صورت است که اسناد توسط چه افرادی، بررسی شده اند. نیاز اصلی این است که یک سند، در یک زمان، تنها توسط یک فرد می تواند بررسی شود. بنابراین در این حالت، رابطه یک تابع است به این دلیل که هر شی موجود در دامنه، تنها به یک شی موجود در برد تابع، نگاشت می شود.

$$
\begin{array}{|l}
\hline
\multicolumn{1}{l}{\textit{Document}} \\
checked\_out : DOCUMENT \pfun PERSON \\
\hline
checked\_out \subseteq permission \\
\hline
\end{array}
$$

شکل ۱۶.۴

$$
\begin{array}{|l}
\multicolumn{1}{l}{\textit{CheckOut}} \\
\hline
\Delta Documents \\
p? : PERSON \\
d? : DOCUMENT \\
\hline
d? \notin \operatorname{dom} checked\_out \\
(d?, p?) \in permission \\
checked\_out' = checked\_out \cup \{(d?, p?)\} \\
\hline
\end{array}
$$

شکل ۱۷.۴

توجه کنید که در شکل ۱۶.۴ ، $checked\_out$ یک تابع جزئی است که با پیکان $\pfun$ مشخص می شود. این بدان معناسب که برخی از اسناد، ممکن است توسط هیچ فردی بررسی نشده باشند. با استفاده از گزاره این مطلب بیان می شود که تنها افرادی که اجازه تغییر سند را دارند، می توانند آن را بررسی کنند.

checked_out={(design، Maryam)، (spec، Maryam)، (code، Ahmad)}

در این مدلسازی، دو عملیات برای تغییر حالت مورد نیاز است که عبارتند از $checkOut$ و $checkIn$. در شکل ۱۷.۴ عملیات $checkOut$ توصیف شده است. این عملیات دو پارامتر ورودی دارد: یکی فرد $p?$ و دیگری سند $d?$.

$checkOut$ دو پیش شرط دارد. اولین پیش شرط، بیان می کند که سند $d?$ تاکنون نباید بررسی شده باشد. این سند نمی تواند در دامنه $checked\_out$ قرار داشته باشد. علاوه بر این، شخص، نیاز به مجوز بررسی دارد: $(d?, p?)$ باید متعلق به $permission$ باشد. اگر این دو پیش شرط

$$
\begin{array}{|l}
\multicolumn{1}{l}{\underline{\quad Unauthorized \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}} \\
\Xi Documents \\
p? : PERSON \\
d? : DOCUMENT \\
\hline
(d?, p?) \notin permission \\
\hline
\end{array}
$$

شکل ۱۸.۴

برآورده شوند، آنگاه می توان زوج مرتب $(d?,p?)$ را به *checked_out* اضافه کرد. این مساله باعث می شود که شخص دیگری نتواند $d?$ را بررسی کند.

در ادامه به بررسی مواردی می پردازیم که در آن پیش شرط ها برآورده نشوند. دو پیش شرط وجود دارد بنابراین دو حالت هم برای برآورده نشدن آنها وجود دارد. یکی اینکه *checked_out* بگوید که این سند پیش از این بررسی شده است : $d? \in dom checked\_out$.

شِمای *Unauthorized* (۱۸.۴) بیان می کند که شخص مجوز دسترسی به سند را ندارد: $(d?,p?) \notin permission$.

در هر دو مورد، حالت سیستم تغییر نمی کند: $\Xi Documents$.

$$CheckedOut \cong [\Xi Cocuments;\ d? : DOCUMENT \mid d? \in \mathrm{dom}\, checked\_out]$$

عملیات کلی *T_CheckOut* سه وضعیت ممکن را پوشش می دهد.

$$T\_CheckOut \cong checkOut \lor CheckedOut \lor Unauthorized$$

این مثال کوچک، ویژگی های خاص مدل های Z را نشان می دهد.

شما می توانید جزئیات را نادیده بگیرید و بر جنبه هایی از مساله که مورد علاقه شماست، تمرکز کنید. در این مثال بیشترین تمرکز بر روی مجوزها و اینکه چه افرادی تاکنون اسناد را بررسی کرده اند، بوده است. در این مثال، مدلسازی از نحوه کپی کردن اسناد، بین مخزن مرکزی و دایرکتوری های محلی کاربران انجام نشده است. در اینجا، مجموعه ای از اسناد و کاربران، به عنوان مجموعه های ثابت مدل شده اند. همچنین مجوزها به عنوان یک ثابت، مدل شده اند. یک سیستم کنترل سند واقعی، باید روش هایی را برای اضافه کردن یک سند جدید و حذف

سندهای قدیمی، فراهم کند. همچنین چنین سیستمی باید امکان انتساب و تغییر مجوزها را نیز داشته باشد. اگر بخواهیم تمام اینها را با استفاده از Z مدل کنیم، باید افراد، اسناد و مجوزها را به عنوان متغیرهای موجود در شِما ارائه دهیم و دیگر این اشیا را نمی توان بصورت انواع پایه و ثوابت سراسری، تعریف کرد.

مدل ها باید تا حد امکان ساده باشند. اگر در ایجاد مدل از توابع و گزاره های بسیار پیچیده استفاده شده است، مسلما روش اشتباهی در پیش گرفته شده است. باید اجازه داد که ویژگی های پایه مجموعه ها، روابط و توابع، کار را انجام دهند. این نیاز که تنها یک نفر در هر زمان بتواند یک سند را بررسی کند، توسط یک تابع قابل تعریف است.

نیازهای مرتبط با مجوزها، با استفاده از تابع $checked\_out$ که زیر مجموعه ای از رابطه $permission$ است، ارائه می شوند. در Z، توابع رابطه هستند و روابط مجموعه هستند بنابراین عملگرهایی که برای مجموعه ها تعریف شده اند، برای روابط و توابع نیز بکار برده می شوند و می توان تمام آنها را با یکدیگر در یک عبارت بکار برد. این یکی از مزایای Z است که در دیگر نشان گذاری های رسمی یافت نمی شود.

۴.۴ نمونه چهارم: ماشین حالت متناهی

در این مثال سه روشی را که برای نمایش مدل یک ماشین حالت متناهی وجود دارد، نشان داده شده است. این سه روش عبارتند از: دیاگرام، جدول و Z.

۱.۴.۴ دیاگرام انتقال حالت

یکی از روش های نمایش ماشین های حالت متناهی، استفاده از دیاگرام انتقال حالت [1] است. در شکل یک مثال از این دیاگرام آورده شده است. حالت ها بصورت دایره هایی نشان داده می شوند؛ انتقال بین حالت ها با استفاده از پیکان مشخص می شود. برچسب هر پیکان رخدادی است که منجر به انتقال از حالت مبدا به حالت مقصد می شود. بنابراین در شکل ، زمانیکه در حالت $PATIENTS$ قرار دارد، با فشار دادن کلید $ENTER$، ماشین به حالت $FIELDS$

[1] State transition diagram

$$
\begin{array}{|l}
no\_change, transitions, control : FSM \\
\hline
control = no\_change \oplus transitions \\
no\_change = \{s : STATE;\ e : EVENT \bullet (s,e) \mapsto s\} \\
transitions = \{(patients, enter) \mapsto fields, \\
\quad (fields, select\_patient) \mapsto patients, (fields, enter) \mapsto setup, \\
\quad (setup, select\_patient) \mapsto patients, (setup, select\_field)) \mapsto fields, (setup, ok) \mapsto ready, \\
\quad (ready, select\_patient) \mapsto patients, (ready, select\_field) \mapsto fields, (ready, start) \mapsto beam\_on, (re \\
\quad (beam\_on, stop) \mapsto ready, (beam\_on, intlk) \mapsto setup\}
\end{array}
$$

شکل ۱۹.۴

منتقل می شود و در حالت *FIELDS* با فشردن کلید *ENTER*، انتقال به حالت *SETUP* رخ می دهد.

می توان دنباله رفتارهای ممکن ماشین را، با دنبال کردن پیکان های ماشین، ردیابی کرد.

۲.۴.۴ جدول انتقال حالت

دیاگرام انتقال حالت، یک تصویر از مدل ماشین حالت متناهی ارائه می دهد. روش های دیگری نیز برای نمایش مدل مشابه وجود دارند. یکی از این روش ها جدول انتقال حالت [1] است.

STATE::=patients | fields | setup | ready | beam_on

EVENT::=select_patient | select_field | enter | start | stop | ok | intlk

$FSM ::= (STATE \times EVENT) \rightarrow\!\!\!\!\!+ STATE$

[1] State transition table

۵

مستندسازی سرویس شبکه با استفاده از Z

در فصل قبل به ارائه تعدادی نمونه ساده که با استفاده از زبان رسمی Z توصیف شده بودند، پرداختیم. در این فصل، چگونگی مستندسازی سرویس های شبکه را به صورت رسمی و با استفاده از زبان Z بررسی خواهیم کرد.

۱.۵ مقدمه

۲.۵ توصیف سرویس

۳.۵ مستندسازی سرویس

۶
واژگان ماشین

۷
توصیف سیستم فایل

در این بخش به ارائه یک مورد مطالعه که از زبان Z استفاده کرده است می پردازیم. ما نشان می دهیم که چگونه شِماها برای توصیف یک سیستم فایل ساده مورد استفاده قرار می گیرند. این شِماها برای نشان دادن ساختمان داده ها و عملیات روی آنها استفاده می شوند. همچنین نشان داده خواهد شد که چگونه پیش شرط های عملیات های مختلف قابل محاسبه است و چگونه می توان توصیف یک فایل تنها را به یک مؤلفه نمایه شده از یک سیستم فایل ارتقا داد.

۱.۷ فاصل برنامه نویسی

فاصل برنامه نویسی برای یک سیستم فایل، دقیقا مشخص می کند که چه چیزی قرار است مدل شود. این فاصل لیستی از عملیات روی سیستم فایل است که با شرح تاثیر آنها کامل شده است. برای مثال: عملیات *create* ممکن است برای تولید یک فایل جدید بکار برده شود و عملیات *read* که برای دسترسی به داده از یک فایل موجود استفاده شود.
عملیات به دو دسته تقسیم شده اند: آنهایی که بر داده های موجود در یک فایل تاثیر می گذارند و آنهایی که بر سیستم فایل بطور کلی تاثیر خواهند گذاشت. در سطح فایل، چهار عملیات زیر

$$
\begin{array}{|l}
\hline
\textit{File} \\
\hline
contents : Key \pfun Data \\
\hline
\end{array}
$$

شکل ۱.۷

$$
\begin{array}{|l}
\hline
\textit{FileInit} \\
\hline
File' \\
\hline
contents' = \varnothing \\
\hline
\end{array}
$$

شکل ۲.۷

وجود دارند:

خواندن [1] : برای خواندن یک بخش از داده، از فایل، استفاده می شود.

نوشتن [2] : برای نوشتن بخش از داده بر روی فایل، استفاده می شود.

اضافه کردن [3] : برای اضافه کردن یک بخش جدید داده به یک فایل استفاده می شود.

حذف کردن [4] : برای حذف بخشی از داده موجود در فایل، استفاده می شود.

عملیات اضافه کردن و نوشتن، با یکدیگر متفاوت اند. عملیات اول این فرصت را فراهم می کند که فایل را با استفاده از داده های جدید گسترش دهیم و عملیات دوم، بخشی از بخش های موجود در فایل را بازنویسی می کند.

رابط برنامه نویسی نیز شامل عملیاتی است که بر روی فایل ها انجام می شوند. در زیر چهار نمونه از این عملیات ها نشان می دهیم:

ایجاد کردن [1] :برای ایجاد یک فایل جدید به کار می رود.

[1] read [2] write [3] add [4] delete [1] create

$$\begin{array}{|l}
\multicolumn{1}{l}{\triangle File} \\ \hline
File \\
File' \\ \hline
\end{array}$$

شکل ۳.۷

$$\begin{array}{|l}
\multicolumn{1}{l}{\Xi File} \\ \hline
\triangle File \\ \hline
\theta File = \theta File' \\ \hline
\end{array}$$

شکل ۴.۷

$$\begin{array}{|l}
\multicolumn{1}{l}{Read_{\circ}} \\ \hline
\Xi File \\
k? : Key \\
d! : Data \\ \hline
k? \in \mathrm{dom}\ \ contents \\
d! = contents\ \ k? \\ \hline
\end{array}$$

شکل ۵.۷

$$\begin{array}{|l}
\multicolumn{1}{l}{Write_{\circ}} \\ \hline
\triangle File \\
k? : Key \\
d! : Data \\ \hline
k? \in \mathrm{dom}\ \ contents \\
contents' = contents \oplus \{k? \mapsto d?\} \\ \hline
\end{array}$$

شکل ۶.۷

$$
\begin{array}{|l}
\multicolumn{1}{l}{\underline{\quad Add_0 \quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad}} \\
\triangle File \\
k? : Key \\
d! : Data \\
\hline
k? \notin \mathbf{dom}\ contents \\
contents' = contents \cup \{k? \mapsto d?\} \\
\hline
\end{array}
$$

شکل ۷.۷

$$
\begin{array}{|l}
\multicolumn{1}{l}{\underline{\quad Delete_0 \quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad}} \\
\delta File \\
k? : Key \\
\hline
k? \in \mathbf{dom}\ contents \\
contents' = \{k?\} \lhd contents \\
\hline
\end{array}
$$

شکل ۸.۷

$$
\begin{array}{|l}
\multicolumn{1}{l}{\underline{\quad KeyError \quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad}} \\
\Xi File \\
k? : Key \\
r! : Report \\
\hline
\end{array}
$$

شکل ۹.۷

$$
\begin{array}{|l}
\hline
\textit{KeyNotInUse} \\
KeyError \\
\hline
k? \notin \mathbf{dom}\ contents \\
r! = key\_in\_use \\
\hline
\end{array}
$$

شکل ۱۰.۷

$$
\begin{array}{|l}
\hline
\textit{KeyInUse} \\
KeyError \\
\hline
k? \in \mathbf{dom}\ contents \\
r! = key\_in\_use \\
\hline
\end{array}
$$

شکل ۱۱.۷

$$
\begin{array}{|l}
\hline
\textit{Success} \\
r! : Report \\
\hline
r! = okay \\
\hline
\end{array}
$$

شکل ۱۲.۷

$$
\begin{array}{|l}
contents, \;\; contents' : Key \nrightarrow Data \\
k? : Key \\
d! : Data \\
r! : Report \\
\hline
(k? \in \mathrm{dom} \;\; contents \;\wedge \\
\;\; d! = contents \;\; k? \;\wedge \\
\;\; contents' = contents \;\wedge \\
\;\; r! = okay) \\
\wedge \\
(k? \notin \mathrm{dom} \;\; contents \;\wedge \\
\;\; contents' = contents \;\wedge \\
\;\; r! = key\_not\_in\_use)
\end{array}
$$

شکل ۱۳.۷

$$
\begin{array}{|l}
\multicolumn{1}{l}{System} \\
\hline
file : Name \nrightarrow File \\
open : \mathbb{P} Name \\
\hline
open \subseteq \mathrm{dom} \;\; file \\
\hline
\end{array}
$$

شکل ۱۴.۷

۲.۷ عملیات بر روی فایل ها

۳.۷ سیستم فایل

۴.۷ تحلیل رسمی

$$
\begin{array}{|l}
\hline
\textit{SystemInit} \\
System' \\
\hline
file' = \varnothing \\
\hline
\end{array}
$$

شکل ۱۵.۷

$$
\begin{array}{|l}
\hline
\textit{Promote} \\
\Delta System \\
\Delta File \\
n? : Name \\
\hline
n? \in open \\
file~~n? = \theta File \\
file'~~n? = \theta File' \\
\{n?\} \mathbin{⩤} file = \{n?\} \mathbin{⩤} file' \\
open' = open \\
\hline
\end{array}
$$

شکل ۱۶.۷

$$
\begin{array}{|l}
\hline
\textit{FileAccess} \\
\Delta System \\
n? : Name \\
\hline
n? \in \mathbf{dom}~~file \\
file' = file \\
\hline
\end{array}
$$

شکل ۱۷.۷

$$
\begin{array}{|l}
\hline
System \\
n? : Name \\
\hline
\exists\, r! : Report \\
\quad n? \notin \operatorname{dom} file \\
\quad r! = file\_does\_not\_exist \\
\hline
\end{array}
$$

شکل ۱۸.۷

$$
\begin{array}{|l}
\hline
System \\
n? : Name \\
\hline
n? \in open \\
\hline
\end{array}
$$

شکل ۱۹.۷

$$
\begin{array}{|l}
\hline
System \\
n? : Name \\
\hline
\end{array}
$$

شکل ۲۰.۷

۸

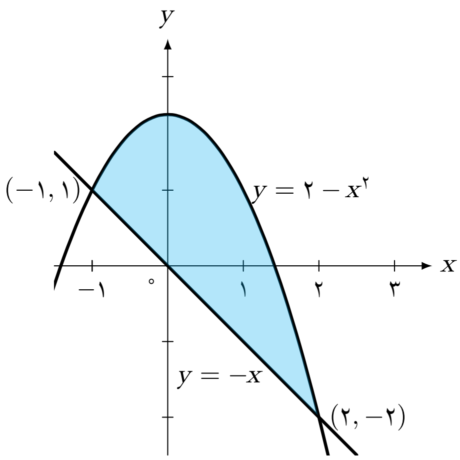
نمونه هایی از کدهای لاتک جهت آموزش

مشتق به طور گسترده در علوم پایه، اقتصاد، پزشکی و علوم کامپیوتر برای محاسبه سرعت اولیه و شتاب و به به منظور توضیح رفتار ماشین‌آلات، تخمین میزان افت آب در هنگام پمپ شدن آب از تانکر آب و پیشگویی نتایج ایجاد خطا در اندازه‌گیری‌ها به کار می‌رود. پیدا کردن مشتق‌ها می‌تواند طولانی و سخت باشد. می‌توان گفت مشتق یکی از ارکان اصلی حساب دیفرانسیل و انتگرال محسوب می‌شود. در این فصل، تکنیک‌هایی برای محاسبه آسان‌تر آن‌ها بیان می‌شود.[؟]

۱.۸ یادآوری حدهای یک‌طرفه و کاربرد آن‌ها

در این بخش ابتدا حدهای یک‌طرفه را یادآوری کرده و بعد از آن، به بیان مفهوم مشتق می‌پردازیم. سپس روابط و قضایای مشتق‌گیری را بیان می‌کنیم. در تعریف $\lim_{x\to a} f(x)$، xهایی را در نظر می‌گیریم که در یک بازهٔ باز شامل a و نه خود a باشند؛ یعنی مقادیر x نزدیک به a را، چه بزرگ‌تر از a باشند و چه کوچک‌تر از آن باشند.

حال فرض کنید تابعی مانند $f(x) = \sqrt{x-۴}$ داریم. چون برای $x < ۴$، مقدار $f(x)$ وجود ندارد، بنابراین f در هیچ بازهٔ باز شامل ۴ تعریف نشده است. لذا $\lim_{x\to ۴} \sqrt{x-۴}$ بی‌معنی است. از آنجایی که استفاده از تعریف مشتق برای مشتق‌گیری از توابع، کاری زمان‌بر است، در این بخش، قضایایی را مطرح می‌کنیم که با کمک آن‌ها بتوان مشتق توابع را به سادگی به دست آورد. شکل ۱.۸ را ببینید که در آن، ناحیهٔ محصور بین نمودار $y = ۲ - x^۲$ و خط $y = -x$ را نشان می‌دهد.



شکل ۱.۸ ناحیهٔ محصور ایجاد شده توسط سهمی $y = ۲ - x^۲$ و خط $y = -x$

با وجود این، اگر x را فقط به مقادیر بزرگ‌تر از ۴ محدود کنیم، می‌توانیم مقدار $\sqrt{x-۴}$ را به اندازهٔ دلخواه به ۰ نزدیک کنیم؛ در چنین حالتی، x را از سمت راست به ۴ میل می‌دهیم و آن‌را حد یک‌طرفه از راست و یا حد راست می‌نامیم. بنابراین

$$\lim_{x\to ۴^+} \sqrt{x-۴} = \circ.$$

حد چپ نیز به صورت مشابه تعریف می‌شود.

اگر تابع f در $x_۱$ تعریف شده باشد، آنگاه مشتق راست f در $x_۱$ با $f'_+(x_۱)$ نشان داده می‌شود و به صورت

$$f'_+(x_۱) = \lim_{\Delta x\to \circ^+} \frac{f(x_۱ + \Delta x) - f(x_۱)}{\Delta x} \qquad (۱.۸)$$

و یا به عبارت دیگر، تعریف می‌شود؛ به شرطی که این حدود موجود باشند. در ادامه، چند قضیه و مثال بیان می‌شود تا خواننده بیشتر با مفاهیم حد و مشتق آشنا شود. برای بحث بیشتر می‌توان به کتاب‌های پیشرفته‌تر حساب دیفرانسیل مراجعه کرد.

قضیه ۱.۱.۸ (وجود حد) گوییم $\lim_{x\to a} f(x)$ وجود دارد و برابر L است، اگر و تنها اگر

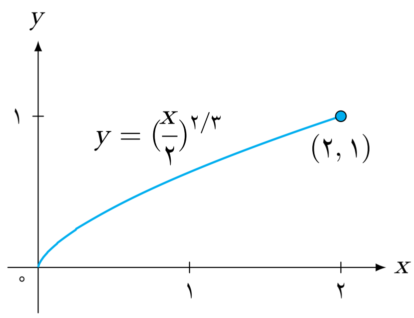
$$\lim_{x\to a^+} f(x), \qquad \lim_{x\to a^-} f(x)$$

هر دو موجود و برابر با L باشند.

مثال ۲.۱.۸. فرض کنید تابع f مطابق شکل ۲.۸ و با ضابطهٔ

$$f(x) = \begin{cases} -۱ & x < ۰ \\ ۰ & x = ۰ \\ ۱ & x > ۰ \end{cases}$$

تعریف شده است. آیا حد[1] تابع f وجود دارد؟



شکل ۲.۸ نمودار تابع $y = (x/۲)^{۲/۳}$ در بازه $[۰, ۲]$

حل. با توجه به تعریف تابع، اگر x عددی کوچک‌تر از ۰ باشد، $f(x)$ دارای مقدار ثابت -۱ است. لذا $\lim_{x\to ۰^-} f(x) = -۱$. با استدلالی مشابه، نتیجه می‌شود که

$$\lim_{x\to ۰^+} f(x) = +۱.$$

[1] در ادامه فرض می‌شود خواننده تا حدودی با مفاهیم پایه‌ای حد آشناست.

بنابراین

$$\lim_{x\to ۰^-} f(x) \neq \lim_{x\to ۰^+} f(x)$$

و لذا حل مثال به پایان می‌رسد. ■

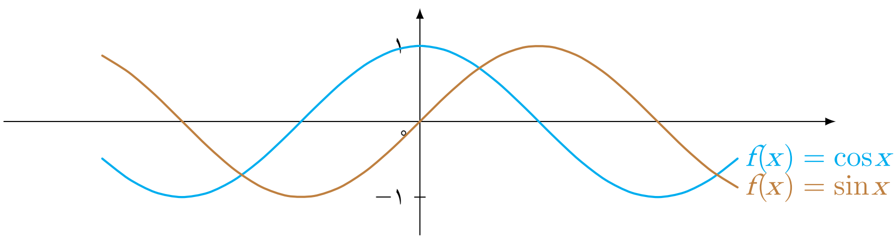
در معنی شناسی نمادین، برنامه‌ها و قطعه‌برنامه‌ها، به عنصرهایی از ساختارهای ریاضی مانند دامنه‌ها از دیدگاه اسکات[۲] نگاشته می‌شود. اگر سیستم مدل‌بندی شده توانایی ایجاد انتخاب‌های تصادفی (یا انتخاب‌های شبه تصادفی) را داشته باشد، آنگاه منطقی است که رفتار خود را به وسیله اندازه‌ای که احتمال را برای سیستم ثبت می‌کند، مدل‌بندی کند تا زیرمجموعه اندازه‌پذیری از مجموعه همه حالت‌های ممکن بشود. این ایده‌ها برای اولین بار توسط صاحب جهرمی[۳] و کازن[۴] مطرح شد. هنگامی که کازن با فضاهای اندازه مطلق کار می‌کرد، اندازه‌های (احتمال) در نظر گرفته شده قبلی، به وسیله مجموعه‌های اسکات‑باز یک dcpo گسترش پیدا کرد.

این ارتباط بین محاسبه‌پذیری و توپولوژی، بطور بسیار واضح توسط اسمیت[۵] شرح داده شده و بعدها توسط آبرامسکی[۶]، ویکرز[۱] و دیگران، بیشتر توسعه داده شد.

تعریف ۳.۱.۸ (مشتق تابع) مشتق تابع f، تابعی است که با علامت f' نشان داده می‌شود و مقدار آن در هر عدد x واقع در دامنهٔ f به صورت

$$f'(x) = \lim_{\Delta x\to ۰} \frac{f(x+\Delta x)-f(x)}{\Delta x} \tag{۲.۸}$$

تعریف می‌شود؛ به شرطی که حد فوق وجود داشته باشد. شکل ۳.۸ را ببینید.



شکل ۳.۸ نمودار دو تابع مثلثاتی $y = \sin x$ و $y = \cos x$

[2]Scott [3]Saheb-Djahromi [4]Kozen [5]Smyth [6]Abramsky [1]Vickers

۲.۸ انتگرال معین و نامعین و کاربرد آن در مهندسی

در این بخش، مفهوم انتگرال‌های معین و نامعین را توضیح داده و سپس خواص انتگرال معین بیان می‌شود. همچنین بعضی از کاربردهای انتگرال معین توضیح داده می‌شود. بعد از آن، نوبت به انتگرال‌های نامعین می‌رسد و روش‌های انتگرال‌گیری برای این نوع انتگرال‌ها شرح داده می‌شود. از این روش‌ها بعدها در درس معادلات دیفرانسیل نیز استفاده می‌شود.

۱.۲.۸ انتگرال معین

فرض کنید $y = f(x)$ یک تابع پیوسته روی بازهٔ $[a,b]$ باشد. این بازه را به n زیربازه با انتخاب $n-1$ نقطه مانند x_1، x_2، $\ldots$، x_{n-1} بین a و b تقسیم می‌کنیم به شرطی که

$$a < x_1 < x_2 < \cdots < x_{n-1} < b.$$

برای ایجاد یکنواختی، a را با x_0 و b را با x_n نشان می‌دهیم. شکل ۴.۸ را ببینید.

قضیه ۱.۲.۸ (قضیه مقدار میانگین برای انتگرال‌های معین). *اگر f روی $[a,b]$ پیوسته باشد، آنگاه یک cای در بازهٔ $[a,b]$ وجود دارد به طوری که*

$$f(c) = \frac{1}{b-a} \int_a^b f(x)dx.$$

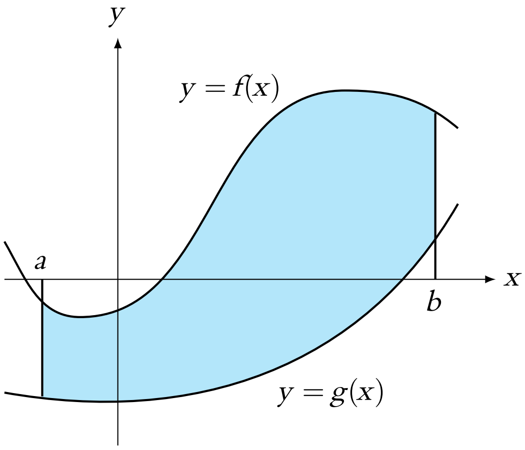
ناحیه ممکن است دارای شکل خاصی باشد که در این صورت، با استفاده از فرمول‌های هندسه [؟] می‌توانیم مساحت آن‌را حساب کنیم.

اگر f و g توابع پیوسته‌ای روی بازهٔ $[a,b]$ و با شرط $f(x) \geq g(x)$ باشند، آنگاه مساحت ناحیهٔ بین منحنی‌های $y = f(x)$ و $y = g(x)$ از a تا b برابر است با

$$A = \int_a^b [f(x) - g(x)]dx. \qquad (۳.۸)$$

۲.۲.۸ منحنی‌های قاطع یکدیگر

وقتی ناحیه‌ای توسط منحنی‌هایی که یکدیگر را قطع می‌کنند، مشخص می‌شود، نقاط تقاطع، حدود انتگرال‌گیری را تعیین می‌کنند. مثال بعدی، نمونه‌ای از این حالت را نشان می‌دهد. در



شکل ۴.۸ ناحیه محصورشده بین دو منحنی $y=f(x)$ و $y=g(x)$

فصل بعدی باز هم نمونه‌های دیگری را بررسی خواهیم کرد که باعث فهم بیشتر مبحث خواهد شد.

با توجه به شکل، طول قطعه‌خط خاص PQ برابر L است. بنابراین طول منحنی به وسیلهٔ جمع

$$\sum_{k=۱}^{n}\sqrt{(\Delta x_k)^۲+(\Delta y_k)^۲}$$

تقریب زده می‌شود.

یادآوری ۱.۸ (محاسبه مشتق) اگر $x_۱$ عدد خاصی از دامنهٔ f باشد، آنگاه می‌توان نوشت

$$f'(x_۱)=\lim_{\Delta x\to ۰}\frac{f(x_۱+\Delta x)-f(x_۱)}{\Delta x} \qquad (۴.۸)$$

البته به شرطی که این حد وجود داشته باشد. از رابطهٔ (۴.۸) برای محاسبهٔ مشتق تابع f در یک نقطهٔ خاص مانند $x_۱$ استفاده می‌شود.

اگر در رابطهٔ (۴.۸) قرار دهیم $x_۱+\Delta x=x$، آنگاه عبارت «$\Delta x\to ۰$» معادل

«$x \to x_1$» است. بنابراین با توجه به فرمول (۴.۸) می‌توان نوشت

$$f'(x_1) = \lim_{x \to x_1} \frac{f(x) - f(x_1)}{x - x_1}$$

تابع f را در x_1 مشتق‌پذیر گوییم، اگر $f'(x_1)$ وجود داشته باشد. تابع f را روی بازهٔ I مشتق‌پذیر گوییم، اگر f به ازای هر عدد واقع در این بازه، مشتق‌پذیر باشد.

مثال ۲.۲.۸. مساحت ناحیهٔ محصور ایجاد شده توسط سهمی $y = ۲ - x^۲$ و خط $y = -x$ را پیدا کنید.

حل. ابتدا نمودار هر دو منحنی را رسم می‌کنیم (شکل ۵.۸). طبق رابطهٔ (۳.۸)، قرار می‌دهیم $f(x) = ۲ - x^۲$ و $g(x) = -x$. حال برای پیدا کردن حدود انتگرال‌گیری، معادلهٔ $۲ - x^۲ = -x$ را حل می‌کنیم. بنابراین $x^۲ - x - ۲ = ۰$. لذا می‌توان نوشت

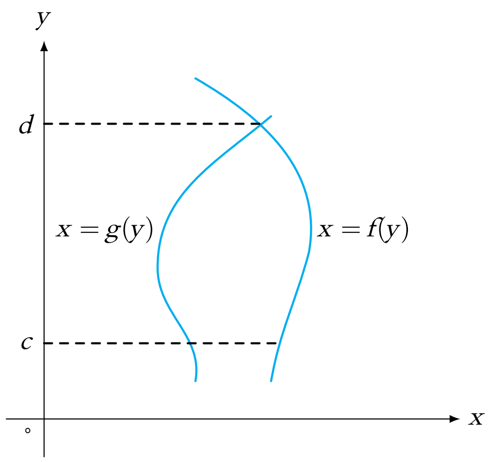
$$\begin{aligned} A &= \cap_a^b [f(x) - g(x)]dx \\ &= \cap_{-۱}^{۲} (۲ + x - x^۲)dx \\ &= \left[۲x + \frac{x^۲}{۲} - \frac{x^۳}{۳}\right]_{-۱}^{۲} \\ &= (۴ + \frac{۴}{۲} - \frac{۸}{۳}) - (-۲ + \frac{۱}{۲} + \frac{۱}{۳}) \end{aligned}$$

که کار را تمام می‌کند. ■

این ایده‌ها برای اولین بار توسط صاحب جهرمی[۲] و کازن[۳] مطرح شد. هنگامی که کازن با فضاهای اندازه مطلق کار می‌کرد، اندازه‌های (احتمال) در نظر گرفته شده قبلی، به وسیله مجموعه‌های اسکات_باز یک dcpo گسترش پیدا کرد.

این ارتباط بین محاسبه‌پذیری و توپولوژی، بطور بسیار واضح توسط اسمیت[۴] شرح داده شده و بعدها توسط آبرامسکی[۵]، ویکرز[۱] و دیگران، بیشتر توسعه داده شد.

[2] Saheb-Djahromi
[3] Lepoldo Smith Kozen
[4] George Smyth
[5] Abramsky
[1] John Vickers



شکل ۵.۸ نمودار منحنی‌های $x = g(y)$ و $x = f(y)$ در بازه $[c,d]$

روش گفته شده در بالا، روش دیسک (شکل ۵.۸) نام دارد. روش دیگری نیز برای محاسبهٔ حجم حاصل از دوران وجود دارد که به روش واشر، معروف شده است . در ادامه بیشتر با این روش آشنا خواهیم شد.[؟]

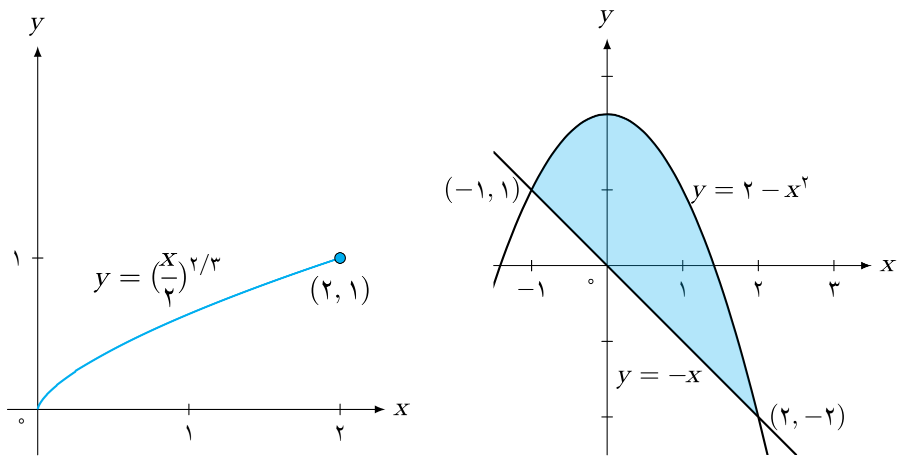
تعریف ۳.۲.۸ (روش واشر) هرگاه ناحیه‌ای که برای تولید یک جسم، دوران داده می‌شود، محور دوران را قطع نکند، جسم تولید شده، دارای یک سوراخ خواهد بود. در این روش، از فرمول

$$V = \int_a^b \pi([R(x)]^{۲} - [r(x)]^{۲})\, dx \qquad (۵.۸)$$

استفاده می‌شود که در آن، $R(x)$ شعاع بیرونی و $r(x)$ شعاع داخلی واشر است.

دقت داشته باشید که در فرمول (۹.۸) اگر $r(x)$ در سراسر بازهٔ $[a,b]$ صفر باشد، همان فرمول روش دیسک، نتیجه می‌شود. بنابراین روش دیسک، حالت خاصی از روش واشر است.

سوالی که ممکن است در اینجا پیش بیاید این است که کدام‌یک از ۳ روش گفته شده، بهتر است؟ واقعیت این است که به طور قطع، نمی‌توان گفت که کدام روش، همیشه بهتر از بقیه عمل می‌کند. جسم‌های حاصل از دوران، جسم‌هایی هستند که شکل آن‌ها از دوران حول محورها به



(آ) ناحیهٔ محصور ایجاد شده توسط سهمی $y = ۲ - x^۲$ و خط $y = -x$

(ب) نمودار تابع $y = (x/۲)^{۲/۳}$ در بازه I

شکل ۶.۸ نمودار تعریف ۳.۲.۸ در حالت متقارن

دست می‌آید.

$$V = \int_c^d \pi([R(y)]^۲ - [r(y)]^۲)\, dy + \int_۰^۱ \pi([\frac{۳}{۲} - \sqrt{y}]^۲)\, dy.$$

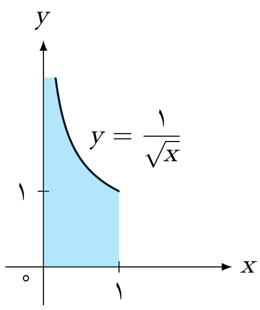
۳.۸ محاسبهٔ طول منحنی‌ها با روشی ابتکاری

فرض کنید می‌خواهیم طول منحنی $y = f(x)$ را از $x = a$ تا $x = b$ پیدا کنیم. طبق معمول، بازهٔ $[a, b]$ را افراز می‌کنیم و نقاط متناظر روی منحنی را با قطعه‌خط‌هایی به همدیگر وصل می‌کنیم تا یک مسیر چندضلعی تشکیل شود (شکل ۷.۸).

تعریف ۱.۳.۸ اگر f روی بازهٔ $[a, b]$ هموار باشد، طول منحنی $y = f(x)$ از a تا b برابر است با

$$L = \int_a^b \sqrt{۱ + (\frac{dy}{dx})^۲}\, dx = \int_a^b \sqrt{۱ + (f'(x))^۲}\, dx. \qquad (۶.۸)$$

گاهی ممکن است dy/dx در یک نقطهٔ خاص از بازهٔ انتگرال‌گیری موجود نباشد. در این حالت dx/dy را حساب می‌کنیم و x را بر حسب تابعی از y بیان می‌کنیم (شکل ۷.۸).



شکل ۷.۸ نمونه‌ای از تابعی با برد نامعین

۴.۸ انتگرال‌های ناسره

انتگرال‌های معینی که تا اینجا با آن‌ها سر و کار داشته‌ایم، دارای دو ویژگی بوده‌اند.[؟] یکی اینکه، دامنهٔ انتگرال‌گیری آن‌ها، یعنی a و b معین بود. دوم اینکه، برد انتگرالده روی این دامنه، معین بود. در این بخش یاد می‌گیریم که چگونه باید با این انتگرال‌ها برخورد کنیم (جدول ۱.۸).

مثال ۱.۴.۸. همگرایی

$$\cap_{۰}^{۳} \frac{dx}{(x-۱)^{۲/۳}}$$

را بررسی کنید.

حل. انتگرالده $f(x) = ۱/(x-۱)^{۲/۳}$ در $x = ۱$ نامتناهی می‌شود؛ اما روی $[۰,۱)$ و $(۱,۳]$ پیوسته است. همگرایی انتگرال روی $[۰,۳]$ به انتگرال‌های از ۰ تا ۱ و ۱ تا ۳ بستگی دارد. روی $[۰,۱)$ داریم

$$\cap_{۰}^{۱} \frac{dx}{(x-۱)^{۲/۳}} = \lim_{b \to ۱^-} \cap_{۰}^{b} \frac{dx}{(x-۱)^{۲/۳}}$$

جدول ۱.۸ نحوه عملکرد تابع f در ارتباط با پیوستگی

نام تابع	نقطه ناپیوستگی	نقطه بحرانی
تابع f	$x=۱$	$a^۲+۳$
تابع g	$x=-۲$	$b-۴$
تابع h	$x=\circ$	$a+b-۷$

$$= \lim_{b\to ۱^-} [۳(b-۱)^{۱/۳} - ۳(\circ-۱)^{۱/۳}]$$
$$= ۳$$

و لذا نتیجه به دست می‌آید. ■

۵.۸ محاسبهٔ حجم جسم‌های حاصل از دوران

جسم‌های حاصل از دوران، جسم‌هایی هستند که شکل آن‌ها از دوران حول محورها به دست می‌آید. گاهی جسم‌های تولید شده، جسم‌هایی هستند که با استفاده از فرمول‌های هندسه، به راحتی می‌توانیم حجم آن‌ها را حساب کنیم؛ اما گاهی شکل این جسم‌ها، منظم نیست و لذا ناچاریم برای محاسبهٔ حجم آن‌ها از حساب دیفرانسیل و انتگرال کمک بگیریم. در ادامه دربارهٔ حجم این نوع جسم‌ها بحث می‌کنیم.

۱.۵.۸ حجم حاصل از دوران حول محور xها

حجم جسم حاصل از دوران ناحیهٔ بین محور xها و نمودار تابع پیوستهٔ $y=R(x)$، $a\leq x\leq b$ حول محور xها برابر است با

$$V = \cap_a^b \pi(R(x))^۲\, dx \qquad (۷.۸)$$

این ناحیه ممکن است دارای شکل خاصی باشد که در این صورت، با استفاده از فرمول‌های هندسه می‌توانیم مساحت آن‌را حساب کنیم؛ اما اگر f و g توابع پیوستهٔ دلخواهی باشند، ناچاریم که مساحت مورد نظر را با استفاده از انتگرال حساب کنیم. حال می‌توان کد این رابطه را به صورت زیر نوشت.

کد ۱.۸ محاسبه حجم جسم حاصل از دوران

```
\def\@makechapterhead#1{%
  \vspace*{50\p@}%
  {\parindent \z@ \raggedright \normalfont
    \ifnum \c@secnumdepth >\m@ne
      \if@mainmatter
        \huge\bfseries \@chapapp\space \thechapter
        \par\nobreak
        \vskip 20\p@
      \fi
    \fi
   \interlinepenalty\@M
  \Huge \bfseries #1\par\nobreak
 \vskip 40\p@
}}
```

۲.۵.۸ حجم حاصل از دوران حول محور yها

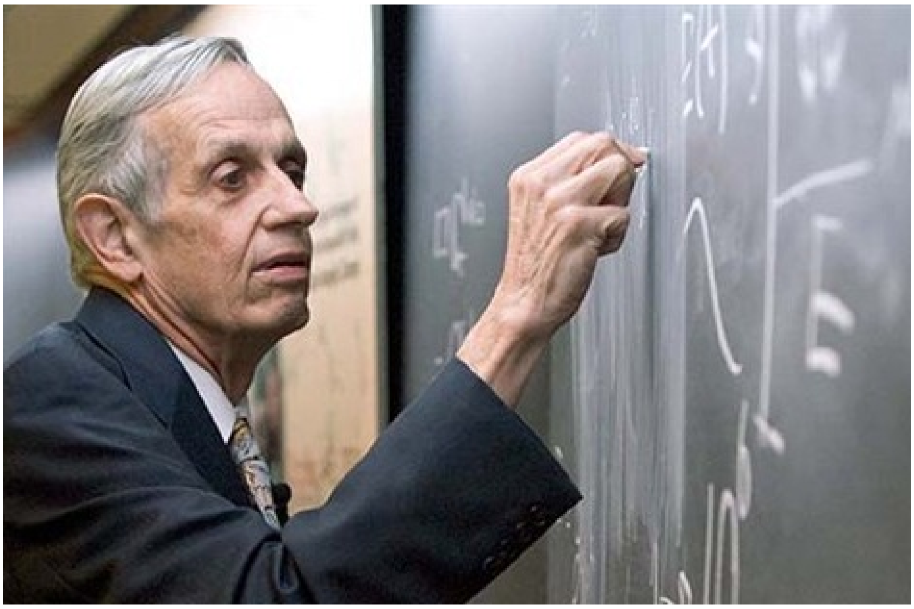
حجم جسم حاصل از دوران ناحیهٔ بین محور yها و نمودار تابع پیوستهٔ $x = R(y)$، $c \leq y \leq d$ حول محور yها برابر است با

$$V = \cap_c^d \pi(R(y))^{۲} dy \tag{۸.۸}$$

حال اگر بتوانیم فرمولی برای طول مسیر ایجاد شده بیابیم، آنگاه فرمولی برای تقریب طول منحنی AB نیز خواهیم داشت.

مثال ۱.۵.۸. مساحت ناحیه‌ای در ربع اول که از بالا به $y = \sqrt{x}$ و از پایین به محور xها و خط $y = x - ۲$ محدود است را بیابید.

حل. ابتدا نمودار هر دو تابع را رسم می‌کنیم. با توجه به شکل بالا، مرز سمت راستی ناحیه، خط $x = y + ۲$ است. گاهی جسم‌های تولید شده، جسم‌هایی هستند که با استفاده از فرمول‌های هندسه، به راحتی می‌توانیم حجم آن‌ها را حساب کنیم؛ لذا $f(y) = y + ۲$ است و مرز $y = -۱$ و $y = ۲$ است. حال چون، مقدار $y = -۱$، یک نقطهٔ تقاطع پایین محور xها را به دست می‌دهد، لذا قابل قبول نیست. بنابراین فقط مقدار $y = ۲$ قابل قبول بوده و لذا $b = ۲$ است.



شکل ۸.۸ جان نش در کلاس درس

حال از رابطهٔ بالا استفاده می‌کنیم.

$$
\begin{aligned}
A = \cap_c^d[f(y) - g(y)]dy &= \cap_{۰}^{۲}[۲ + y - y^{۲}]dy \\
&= \left[۲y + \frac{y^{۲}}{۲} - \frac{y^{۳}}{۳}\right]_{۰}^{۲} \\
&= \frac{۱۰}{۳}.
\end{aligned}
$$

بنابراین $A = ۱۰/۳$ است. ■

هرگاه ناحیه‌ای که برای تولید یک جسم، دوران داده می‌شود، محور دوران را قطع نکند، جسم تولید شده، دارای یک سوراخ خواهد بود. در این روش، از فرمول

$$V = \cap_a^b \pi([R(x)]^{۲} - [r(x)]^{۲})\, dx \qquad (۹.۸)$$

استفاده می‌شود که در آن، $R(x)$ شعاع بیرونی و $r(x)$ شعاع داخلی واشر است.

همان‌طور که دیده می‌شود، نتیجهٔ به دست آمده، با نتیجهٔ مثال قبل یکسان است و با مقدار محاسبات کمتری به دست آمده است. همچنین دقت شود که در این مثال، چون نسبت به y انتگرال گرفته‌ایم، تنها یک انتگرال‌گیری لازم است.

۶.۸ قواعد انتگرال‌گیری نامعین

حجم جسم حاصل از دوران ناحیهٔ بین محور xها و نمودار تابع پیوستهٔ $y = R(x)$، $a \leq x \leq b$ حول محور xها برابر است با

$$V = \cap_a^b \pi (R(x))^2 \, dx \qquad (۱۰.۸)$$

مثال ۱.۶.۸. ناحیهٔ بین منحنی $y = \sqrt{x}$، $0 \leq x \leq 4$ و محور xها، برای تولید جسمی، حول محور xها دوران داده می‌شود. حجم آن‌را را پیدا کنید.

حجم جسم حاصل از دوران ناحیهٔ بین محور yها و نمودار تابع پیوستهٔ $x = R(y)$، $c \leq y \leq d$ حول محور yها برابر است با

$$V = \cap_c^d \pi (R(y))^2 \, dy \qquad (۱۱.۸)$$

۷.۸ تکنیک‌های انتگرال‌گیری

۱.۷.۸ انتگرال‌گیری جزء به جزء

انتگرال‌گیری جزء به جزء یکی از تکنیک‌هایی است که برای ساده کردن انتگرال‌هایی به فرم

$$\cap f(x) g(x) dx \qquad (۱۲.۸)$$

به کار می‌رود؛ به شرطی که در آن، از f بتوان بارها مشتق گرفت و از g نیز بتوان به سادگی، انتگرال گرفت. انتگرال

$$\cap x e^x dx,$$

یک نمونه از چنین انتگرالی است؛ زیرا از $f(x) = x$ می‌توان دو بار مشتق گرفت تا صفر شود و

از $g(x) = e^x$ نیز می‌توان به سادگی، بارها انتگرال گرفت. انتگرال‌گیری جزء به جزء، همچنین برای انتگرال‌هایی مانند

$$\cap e^x \sin x dx$$

که در آن‌ها، هر قسمت از انتگرالده بعد از مشتق‌گیری و یا انتگرال‌گیری مکرر، دوباره تکرار می‌شوند، نیز به کار می‌رود.

۲.۷.۸ جانشینی ساده‌کننده

گوییم تابع $F(x)$ یک ضدمشتق تابع $f(x)$ است، هر گاه برای هر x در دامنهٔ f داشته باشیم

$$F'(x) = f(x).$$

مجموعهٔ تمام ضدمشتق‌های f، انتگرال نامعین f نسبت به x نامیده شده و با علامت

$$\cap f(x)dx$$

نشان داده می‌شود.

۳.۷.۸ کسرهای جزیی

طبق قضیهٔ مقدار میانگین، می‌دانیم توابعِ با مشتق یکسان، در یک عدد ثابت با یکدیگر اختلاف دارند. به عبارت دیگر، اگر دو تابع f و g، مشتق یکسانی داشته باشند، آنگاه $f(x) = g(x) + C$ است. بنابراین می‌توان نتیجه گرفت که هر گاه یک ضدمشتق F برای تابع f پیدا کنیم، ضدمشتق‌های دیگر f در یک ثابت، با F تفاوت دارند. می‌توان نتیجه گرفت که هر گاه یک ضدمشتق F برای تابع f پیدا کنیم، ضدمشتق‌های دیگر f در یک ثابت، با F تفاوت دارند. این حالت را در انتگرال‌گیری با

$$\cap f(x)dx = F(x) + C$$

نشان می‌دهیم.

۸.۸ ظاهر شدن انتگرال اصلی در فرایند انتگرال‌گیری

در این بخش چند حکم را با هم مرور می‌کنیم.

لم ۱.۸.۸. *مقدار* $\cap_{۰}^{۱}\sqrt{۱+\cos x}dx$ *نمی‌تواند ۲ باشد.*

Proof

به دلیل سادگی برهان، به عنوان تمرین به خواننده واگذار می‌شود.

گزاره ۲.۸.۸. *مقدار* $\cap_{۰}^{۱}\sqrt{۱+\cos x}dx$ *نمی‌تواند ۲ باشد.*

نتیجه ۳.۸.۸. *مقدار* $\cap_{۰}^{۱}\sqrt{۱+\cos x}dx$ *نمی‌تواند ۲ باشد.*

ملاحظه ۴.۸.۸. *مقدار* $\cap_{۰}^{۱}\sqrt{۱+\cos x}dx$ *نمی‌تواند ۲ باشد.*

اگر $f(x) = ۳x^{۲} + ۱۲$ باشد، مشتق آن‌را حساب کنید. اگر x عددی در دامنهٔ f باشد، با استفاده از مطالب قبلی داریم

$$\begin{aligned}
f'(x) &= \lim_{\Delta x\to ۰} \frac{f(x+\Delta x)-f(x)}{\Delta x} \\
&= \lim_{\Delta x\to ۰} \frac{۳(x+\Delta x)^{۲}+۱۲-(۳x^{۲}+۱۲)}{\Delta x} \\
&= \lim_{\Delta x\to ۰} \frac{۳x^{۲}+۶x\Delta x+۳(\Delta x)^{۲}+۱۲-۳x^{۲}-۱۲}{\Delta x} \\
&= \lim_{\Delta x\to ۰} \frac{۶x\Delta x+۳(\Delta x)^{۲}}{\Delta x} \\
&= \lim_{\Delta x\to ۰} ۶x+۳(\Delta x) \\
&= ۶x
\end{aligned}$$

و لذا مشتق تابع f به دست می‌آید. اگر x_1 عدد خاصی از دامنهٔ f باشد، آنگاه می‌توان نوشت

$$f'(x_1) = \lim_{\Delta x \to 0} \frac{f(x_1 + \Delta x) - f(x_1)}{\Delta x} \tag{۱۳.۸}$$

البته به شرطی که این حد وجود داشته باشد. از رابطهٔ (؟؟) برای محاسبهٔ مشتق تابع f در یک نقطهٔ خاص مانند x_1 استفاده می‌شود.

اگر در رابطهٔ (۱۳.۸) قرار دهیم $x = x_1 + \Delta x$، آنگاه عبارت «$\Delta x \to 0$» معادل «$x \to x_1$» است. بنابراین با توجه به فرمول (۴.۸) می‌توان نوشت

$$f'(x_1) = \lim_{x \to x_1} \frac{f(x) - f(x_1)}{x - x_1} \tag{۱۴.۸}$$

مشتق تابع $f(x) = 3x^2 + 12$ را در نقطهٔ $x = 2$ حساب کنید.

و لذا مشتق تابع f در نقطهٔ $x = 2$ به دست می‌آید. تابع f را در x_1 مشتق‌پذیر گوییم، اگر $f'(x_1)$ وجود داشته باشد. تابع f را روی بازهٔ I مشتق‌پذیر گوییم، اگر f به ازای هر عدد واقع در این بازه، مشتق‌پذیر باشد. اگر $f(x) = 3x^2 + 12$ باشد، تعیین کنید که f در کجا مشتق‌پذیر است؟ از آنجایی که $f'(x) = 6x$ و $6x$ برای تمام اعداد حقیقی موجود است، لذا نتیجه می‌شود که f در همه جا مشتق‌پذیر است. اگر تابع f در x_1 تعریف شده باشد، آنگاه مشتق راست f در x_1 با $f'_+(x_1)$ نشان داده می‌شود و به صورت

$$f'_+(x_1) = \lim_{\Delta x \to 0^+} \frac{f(x_1 + \Delta x) - f(x_1)}{\Delta x} \tag{۱۵.۸}$$

و یا به عبارت دیگر،

$$f'_+(x_1) = \lim_{x \to x_1^+} \frac{f(x) - f(x_1)}{x - x_1} \tag{۱۶.۸}$$

تعریف می‌شود؛ به شرطی که این حدود موجود باشند. اگر تابع f در x_1 تعریف شده باشد، آنگاه مشتق چپ f در x_1 با $f'_-(x_1)$ نشان داده می‌شود و به صورت

$$f'_-(x_1) = \lim_{\Delta x \to 0^-} \frac{f(x_1 + \Delta x) - f(x_1)}{\Delta x} \tag{۱۷.۸}$$

و یا به عبارت دیگر،

$$f'_-(x_1) = \lim_{x \to x_1^-} \frac{f(x) - f(x_1)}{x - x_1} \qquad (۱۸.۸)$$

تعریف می‌شود؛ به شرطی که این حدود موجود باشند. فرض کنید تابع f به صورت

$$f(x) = \begin{cases} ۲x - ۱ & x < ۳ \\ ۸ - x & ۳ \leq x \end{cases}$$

تعریف شده است. پیوستگی و مشتق‌پذیری این تابع را در نقطهٔ $x = ۳$ بررسی کنید. برای بررسی پیوستگی، سه شرط پیوستگی را بررسی می‌کنیم. (۱) داریم $f(۳) = ۵$. بنابراین شرط اول که همان موجود بودن $f(۳)$ است، برقرار است. (۲) برای بررسی شرط دوم داریم

$$\lim_{x \to ۳^-} f(x) = \lim_{x \to ۳^-} (۲x - ۱) = ۵$$

و

$$\lim_{x \to ۳^+} f(x) = \lim_{x \to ۳^+} (۸ - x) = ۵.$$

بنابراین $\lim_{x \to ۳} f(x) = ۵$ و لذا شرط دوم هم برقرار است. (۳) $\lim_{x \to ۳} f(x) = f(۳)$. بنابراین f در ۳ پیوسته است. حال مشتق‌پذیری f را در ۳ بررسی می‌کنیم. داریم

$$\begin{aligned} f'_-(۳) &= \lim_{\Delta x \to ۰^-} \frac{f(۳ + \Delta x) - f(۳)}{\Delta x} \\ &= \lim_{\Delta x \to ۰^-} \frac{(۲(۳ + \Delta x) - ۱) - ۵}{\Delta x} \\ &= \lim_{\Delta x \to ۰^-} \frac{۶ + ۲\Delta x - ۶}{\Delta x} \\ &= ۲. \end{aligned}$$

۹.۸ سه جانشینی بنیادی

تابع f را در x_1 مشتق‌پذیر گوییم، اگر $f'(x_1)$ وجود داشته باشد. تابع f را روی بازهٔ I مشتق‌پذیر گوییم، اگر f به ازای هر عدد واقع در این بازه، مشتق‌پذیر باشد.

جان اسمیت در کتابش می‌نویسد:

طبق قضیهٔ مقدار میانگین، می‌دانیم توابعِ با مشتق یکسان، در یک عدد ثابت با یکدیگر اختلاف دارند. به عبارت دیگر، اگر دو تابع f و g، مشتق یکسانی داشته باشند، آنگاه $f(x) = g(x) + C$ است. بنابراین می‌توان نتیجه گرفت که هر گاه یک ضدمشتق F برای تابع f پیدا کنیم، ضدمشتق‌های دیگر f در یک ثابت، با F تفاوت دارند.

فرض کنید می‌خواهیم طول منحنی $y = f(x)$ را از $x = a$ تا $x = b$ پیدا کنیم. طبق معمول، بازهٔ $[a, b]$ را افراز می‌کنیم و نقاط متناظر روی منحنی را با قطعه‌خط‌هایی به همدیگر وصل می‌کنیم تا یک مسیر چندضلعی تشکیل شود.

تمرین‌ها

۱.۸ اگر $f(x) = 3x^2 + 12$ باشد، مشتق آن‌را حساب کنید.

۲.۸ نشان دهید مقدار

$$\cap_0^1 \sqrt{1 + \cos x}\,dx$$

نمی‌تواند ۲ باشد.

۳.۸ از نامساوی $\cos x \geq (1 - x^2/2)$ که برای هر xی برقرار است، استفاده کنید و یک کران پایین برای مقدار $\cap_0^1 \cos x\,dx$ پیدا کنید.

۴.۸ مشتق تابع $f(x) = 3x^2 + 12$ را در نقطهٔ $x = 2$ حساب کنید.

۵.۸ اگر $f(x) = 3x^2 + 12$ باشد، تعیین کنید که f در کجا مشتق‌پذیر است؟

۶.۸ متحرکی روی نمودار $y = \sqrt{x - 2}$ با فرض $x \geq 2$ حرکت می‌کند. اگر مؤلفهٔ x آن با آهنگ ۲ متر بر ثانیه افزایش یابد، در لحظه‌ای که $x = 6$ است، آهنگ تغییر مؤلفه y آن برابر چیست؟ آیا متحرک در حال صعود است یا نزول؟

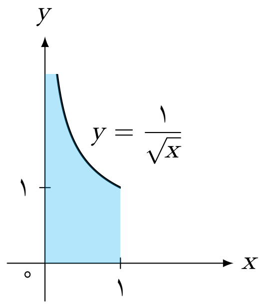
۷.۸ مشتق تابع $y = (\sqrt{x} + 1)(\sqrt{x} - 1)(x + 1)$ را حساب کنید.

۸.۸ مشتق

$$f(x) = \frac{(x - 1)(x^2 - 2x)}{x^4}$$

را حساب کنید.

۹.۸ طول منحنی زیر را چطور می‌توان محاسبه کرد؟



۱۰.۸ اگر $f(x)$ تابعی باشد به طوری که $f(۴) = -۳$ و $f'(۴) = -۵$ و g تابعی باشد به طوری که $g(x) = f(x)/x$ باشد، $g'(۴)$ را به دست آورید.

۱۱.۸ اگر تابع f به صورت

$$y = \sin x - ۲\cos x \tag{۱۹.۸}$$

داده شده باشد، رابطه‌ای بین y و y' بیابید که به x بستگی نداشته باشد.

۱۲.۸ مشتق تابع $y = \sqrt{x^۲ + ۱}$ را حساب کنید.

۱۳.۸ اگر

$$\lim_{h\to o} \frac{f(۳+h) - f(۳)}{h} = -۱$$

باشد، مشتق $y = f(x^۴ + x + ۱)$ را در نقطهٔ $x = ۱$ حساب کنید.

۱۴.۸ اگر

$$f(\sin x - \cos x) = \sqrt{۲}g(۲x - \frac{\pi}{۴})$$

و $g'(\frac{\pi}{۴}) = ۴$ باشد، $f'(۰)$ را حساب کنید.

۱۵.۸ در معادلهٔ زیر، dy/dx را به دست آورید.

$$۲y = x^۲ + \sin y.$$

۱۶.۸ مشتق معادلهٔ پارامتری

$$x = ۳t^۴ + t^۲ - ۵, \quad y = ۶t^۲ - t$$

را به دست آورید.

۱۷.۸ فرض کنید f و g توابع حقیقی و مشتق‌پذیر باشند و $f(۰) = g(۰) = ۰$ و $g'(x) \neq ۰$ باشد. ثابت کنید

$$\lim_{x\to ۰} \frac{f(x)}{g(x)} = \frac{f'(۰)}{g'(۰)}.$$

۱۸.۸ مشتق تابع $y = \mathrm{Arcsin}\, x$ را به دست آورید.

۱۹.۸ شتق تابع $y = \mathrm{Arctan}\, x$ را به دست آورید.

۲۰.۸ معادلهٔ خط مماس بر منحنی

$$y = \mathrm{Arcsin}\, \frac{x - ۱}{x + ۲}$$

را در نقطهٔ تلاقی منحنی با محور yها را بنویسید.

۲۱.۸ در تابع

$$f(x) = \frac{۵x + ۱}{x - ۱},$$

مقدار $(f^{-۱})'(۱۱)$ را به دست آورید.

۲۲.۸ در تابع

$$y = \sqrt{\frac{۱}{\cos ۲x}},$$

رابطه‌ای بین y، y' و y'' بیابید که مستقل از x باشد.

۲۳.۸ در تابع

$$y = \frac{۱}{\sqrt{x^۲ + ۲x + ۵}},$$

رابطه‌ای بین y، y' و y'' بیابید که مستقل از x باشد.

تمرین‌ها

۱.۸ اگر

$$\lim_{h\to o}\frac{f(۳+h)-f(۳)}{h}=-۱$$

باشد، مشتق $y=f(x^۴+x+۱)$ را در نقطهٔ $x=۱$ حساب کنید.

۲.۸ اگر

$$f(\sin x-\cos x)=\sqrt{۲}g(۲x-\frac{\pi}{۴})$$

و $g'(\frac{\pi}{۴})=۴$ باشد، $f'(۰)$ را حساب کنید.

۳.۸ در معادلهٔ زیر، dy/dx را به دست آورید.

$$۲y=x^۲+\sin y.$$

۴.۸ مشتق معادلهٔ پارامتری

$$x=۳t^۴+t^۲-۵,\quad y=۶t^۲-t$$

را به دست آورید.

۵.۸ فرض کنید f و g توابع حقیقی و مشتق‌پذیر باشند و $f(۰)=g(۰)=۰$ و $g'(x)\neq ۰$ باشد. ثابت کنید

$$\lim_{x\to ۰}\frac{f(x)}{g(x)}=\frac{f'(۰)}{g'(۰)}.$$

پیوست آ
چند یادآوری اساسی

جسم‌های حاصل از دوران، جسم‌هایی هستند که شکل آن‌ها از دوران حول محورها به دست می‌آید. گاهی جسم‌های تولید شده، جسم‌هایی هستند که با استفاده از فرمول‌های هندسه، به راحتی می‌توانیم حجم آن‌ها را حساب کنیم؛ اما گاهی شکل این جسم‌ها، منظم نیست و لذا ناچاریم برای محاسبهٔ حجم آن‌ها از حساب دیفرانسیل و انتگرال کمک بگیریم. ناحیه ممکن است دارای شکل خاصی باشد که در این صورت، با استفاده از فرمول‌های هندسه می‌توانیم مساحت آن‌را حساب کنیم.

آ.۱ استقرای ریاضی و چند مثال

وقتی ناحیه‌ای توسط منحنی‌هایی که یکدیگر را قطع می‌کنند، مشخص می‌شود، نقاط تقاطع، حدود انتگرال‌گیری را تعیین می‌کنند. مثال بعدی، نمونه‌ای از این حالت را نشان می‌دهد. سوالی که ممکن است در اینجا پیش بیاید این است که کدام‌یک از ۳ روش گفته شده، بهتر است؟ واقعیت این است که به طور قطع، نمی‌توان گفت که کدام روش، همیشه بهتر از بقیه عمل می‌کند. بنابراین در هر مساله، باید ابتدا ناحیه مورد نظر را رسم کرده و سپس با توجه به آن، بهترین روش را انتخاب کنیم.

آ.۲ مشتق‌های جزئی

تابع f را در $x_۱$ مشتق‌پذیر گوییم، اگر $f'(x_۱)$ وجود داشته باشد. تابع f را روی بازهٔ I مشتق‌پذیر گوییم، اگر f به ازای هر عدد واقع در این بازه، مشتق‌پذیر باشد. اگر فرض کنیم $f(x) = ۳x^۲+۱۲$ باشد، مشتق آن‌را حساب کنید. اگر x عددی در دامنهٔ f باشد، با استفاده از مطالب قبلی داریم

$$\begin{aligned}
f'(x) &= \lim_{\Delta x \to ۰} \frac{f(x+\Delta x) - f(x)}{\Delta x} \\
&= \lim_{\Delta x \to ۰} \frac{۳(x+\Delta x)^۲ + ۱۲ - (۳x^۲ + ۱۲)}{\Delta x} \\
&= \lim_{\Delta x \to ۰} \frac{۳x^۲ + ۶x\Delta x + ۳(\Delta x)^۲ + ۱۲ - ۳x^۲ - ۱۲}{\Delta x} \\
&= \lim_{\Delta x \to ۰} \frac{۶x\Delta x + ۳(\Delta x)^۲}{\Delta x} \\
&= \lim_{\Delta x \to ۰} ۶x + ۳(\Delta x) \\
&= ۶x
\end{aligned}$$

و لذا مشتق تابع f به دست می‌آید.

آ.۳ بسط تیلور

ناحیه ممکن است دارای شکل خاصی باشد که در این صورت، با استفاده از فرمول‌های هندسه می‌توانیم مساحت آن‌را حساب کنیم. حجم جسم حاصل از دوران ناحیهٔ بین محور xها و نمودار تابع پیوستهٔ $y = R(x)$، $a \le x \le b$ حول محور xها برابر است با

$$V = \cap_a^b \pi (R(x))^۲ \, dx \qquad (آ.۱)$$

مثال آ.۳.۱. ناحیهٔ بین منحنی $y = \sqrt{x}$، $۰ \le x \le ۴$ و محور xها، برای تولید جسمی، حول محور xها دوران داده می‌شود. حجم آن‌را را پیدا کنید.

فرض کنید $y = f(x)$ یک تابع پیوسته روی بازهٔ $[a,b]$ باشد. این بازه را به n زیربازه با انتخاب $n - ۱$ نقطه مانند $x_۱$، $x_۲$، ...، $x_{n-۱}$ بین a و b تقسیم می‌کنیم. حجم جسم حاصل از دوران ناحیهٔ بین محور yها و نمودار تابع پیوستهٔ $x = R(y)$، $c \leq y \leq d$ حول محور yها برابر است با

$$V = \cap_c^d \pi (R(y))^۲ \, dy \tag{آ.۲}$$

آ.۴ مختصات قطبی

اگر در رابطهٔ (آ.۲) قرار دهیم $x = x_۱ + \Delta x$، آنگاه عبارت «$\Delta x \to ۰$» معادل «$x \to x_۱$» است. بنابراین با توجه به فرمول (آ.۲) می‌توان نوشت

$$f'(x_۱) = \lim_{x \to x_۱} \frac{f(x) - f(x_۱)}{x - x_۱} \tag{آ.۳}$$

مشتق تابع $f(x) = ۳x^۲ + ۱۲$ را در نقطهٔ $x = ۲$ حساب کنید. لذا مشتق تابع f در نقطهٔ $x = ۲$ به دست می‌آید.

قضیه آ.۴.۱ طبق قضیهٔ مقدار میانگین، می‌دانیم توابعِ با مشتق یکسان، در یک عدد ثابت با یکدیگر اختلاف دارند. به عبارت دیگر، اگر دو تابع f و g، مشتق یکسانی داشته باشند، آنگاه $f(x) = g(x) + C$ است. بنابراین می‌توان نتیجه گرفت که هر گاه یک ضدمشتق F برای تابع f پیدا کنیم، ضدمشتق‌های دیگر f در یک ثابت، با F تفاوت دارند. می‌توان نتیجه گرفت که هر گاه یک ضدمشتق F برای تابع f پیدا کنیم، ضدمشتق‌های دیگر f در یک ثابت، با F تفاوت دارند. این حالت را در انتگرال‌گیری با

$$\cap f(x) dx = F(x) + C$$

نشان می‌دهیم. وقتی ناحیه‌ای توسط منحنی‌هایی که یکدیگر را قطع می‌کنند، مشخص می‌شود، نقاط تقاطع، حدود انتگرال‌گیری را تعیین می‌کنند. مثال بعدی، نمونه‌ای از این حالت را نشان می‌دهد. سوالی که ممکن است در اینجا پیش بیاید این است که کدام‌یک از ۳ روش گفته

شده، بهتر است؟ واقعیت این است که به طور قطع، نمی‌توان گفت که کدام روش، همیشه بهتر از بقیه عمل می‌کند. بنابراین در هر مساله، باید ابتدا ناحیه مورد نظر را رسم کرده و سپس با توجه به آن، بهترین روش را انتخاب کنیم.

تابع f را در $x_۱$ مشتق‌پذیر گوییم، اگر $f'(x_۱)$ وجود داشته باشد. تابع f را روی بازهٔ I مشتق‌پذیر گوییم، اگر f به ازای هر عدد واقع در این بازه، مشتق‌پذیر باشد. اگر $f(x) = ۳x^۲ + ۱۲$ باشد، تعیین کنید که f در کجا مشتق‌پذیر است؟ از آنجایی که $f'(x) = ۶x$ و $۶x$ برای تمام اعداد حقیقی موجود است، لذا نتیجه می‌شود که f در همه جا مشتق‌پذیر است. اگر تابع f در $x_۱$ تعریف شده باشد، آنگاه مشتق راست f در $x_۱$ با $f'_+(x_۱)$ نشان داده می‌شود و به صورت

$$f'_+(x_۱) = \lim_{\Delta x \to ۰^+} \frac{f(x_۱ + \Delta x) - f(x_۱)}{\Delta x} \tag{آ.۴}$$

و یا به عبارت دیگر،

$$f'_+(x_۱) = \lim_{x \to x_۱^+} \frac{f(x) - f(x_۱)}{x - x_۱} \tag{آ.۵}$$

تعریف می‌شود؛ به شرطی که این حدود موجود باشند. اگر تابع f در $x_۱$ تعریف شده باشد، آنگاه مشتق چپ f در $x_۱$ با $f'_-(x_۱)$ نشان داده می‌شود و به صورت

$$f'_-(x_۱) = \lim_{\Delta x \to ۰^-} \frac{f(x_۱ + \Delta x) - f(x_۱)}{\Delta x} \tag{آ.۶}$$

و یا به عبارت دیگر،

$$f'_-(x_۱) = \lim_{x \to x_۱^-} \frac{f(x) - f(x_۱)}{x - x_۱} \tag{آ.۷}$$

تعریف می‌شود؛ به شرطی که این حدود موجود باشند.

آ.۵ بردارها در فضا و خواص آن‌ها

در این بخش چند حکم را با هم مرور می‌کنیم. مقدار $\cap_۰^۱ \sqrt{۱ + \cos x}dx$ نمی‌تواند ۲ باشد. به دلیل سادگی برهان، به عنوان تمرین به خواننده واگذار می‌شود.

اگر $f(x) = ۳x^۲ + ۱۲$ باشد، مشتق آن‌را حساب کنید. اگر $x_۱$ عدد خاصی از دامنهٔ f باشد، آنگاه می‌توان نوشت

$$f'(x_۱) = \lim_{\Delta x \to ۰} \frac{f(x_۱ + \Delta x) - f(x_۱)}{\Delta x} \qquad (\text{آ.۸})$$

البته به شرطی که این حد وجود داشته باشد. از رابطهٔ (آ.۸) برای محاسبهٔ مشتق تابع f در یک نقطهٔ خاص مانند $x_۱$ استفاده می‌شود.

فرض کنید تابع f به صورت

$$f(x) = \begin{cases} ۲x - ۱ & x < ۳ \\ ۸ - x & ۳ \le x \end{cases}$$

تعریف شده است. پیوستگی و مشتق‌پذیری این تابع را در نقطهٔ $x = ۳$ بررسی کنید. برای بررسی پیوستگی، سه شرط پیوستگی را بررسی می‌کنیم. (۱) داریم $f(۳) = ۵$. بنابراین شرط اول که همان موجود بودن $f(۳)$ است، برقرار است. (۲) برای بررسی شرط دوم داریم

$$\lim_{x \to ۳^-} f(x) = \lim_{x \to ۳^-} (۲x - ۱) = ۵$$

و

$$\lim_{x \to ۳^+} f(x) = \lim_{x \to ۳^+} (۸ - x) = ۵.$$

بنابراین $\lim_{x \to ۳} f(x) = ۵$ و لذا شرط دوم هم برقرار است. (۳) $\lim_{x \to ۳} f(x) = f(۳)$. بنابراین f در ۳ پیوسته است. حال مشتق‌پذیری f را در ۳ بررسی می‌کنیم. داریم

$$\begin{aligned} f'_-(۳) &= \lim_{\Delta x \to ۰^-} \frac{f(۳ + \Delta x) - f(۳)}{\Delta x} \\ &= \lim_{\Delta x \to ۰^-} \frac{(۲(۳ + \Delta x) - ۱) - ۵}{\Delta x} \\ &= \lim_{\Delta x \to ۰^-} \frac{۶ + ۲\Delta x - ۶}{\Delta x} \\ &= \lim_{\Delta x \to ۰^-} \frac{۲\Delta x}{\Delta x} \\ &= \lim_{\Delta x \to ۰^-} ۲ \\ &= ۲. \end{aligned}$$

آ.۶ ضرب برداری

تابع f را در x_1 مشتق‌پذیر گوییم، اگر $f'(x_1)$ وجود داشته باشد. تابع f را روی بازهٔ I مشتق‌پذیر گوییم، اگر f به ازای هر عدد واقع در این بازه، مشتق‌پذیر باشد.هر گاه یک ضدمشتق F برای تابع f پیدا کنیم، ضدمشتق‌های دیگر f در یک ثابت، با F تفاوت دارند. می‌توان نتیجه گرفت که هر گاه یک ضدمشتق F برای تابع f پیدا کنیم، ضدمشتق‌های دیگر f در یک ثابت، با F تفاوت دارند. این حالت را در انتگرال‌گیری با

$$\cap f(x)dx = F(x) + C$$

نشان می‌دهیم.

کد آ.۱ روش به دست آوردن انتگرال نامعین

```
\newcommand{\@tufte@lof@line}[2]{%
  \leftskip 0.0em
  \rightskip 0em
  \parfillskip 0em plus 1fil
  \parindent 0.0em
  \@afterindenttrue
  \interlinepenalty\@M
  \leavevmode
  \@tempdima 2.0em
  \advance\leftskip\@tempdima
  \null\nobreak\hskip -\leftskip
  {#1}\nobreak\qquad\nobreak#2%
  \par%
}
```

پاسخ تمرین‌های برگزیده

فصل ۱

۱.۸ $6x$

۳.۸ 2

۴.۸ با مشتق‌گیری داریم $f'(2) = 6(2) = 12$

۵.۸ تابع f در همه جا مشتق‌پذیر است.

۶.۸ -6. نزول می‌کند.

۷.۸ ابتدا عبارت را ساده می‌کنیم:

$$y = (x-1)(x+1) = x^2 - 1$$

بنابراین $y' = 2x$.

۱۰.۸ $1/2$

۱۳.۸ بعد از ساده کردن عبارت، نتیجه می‌شود $y' = 3$.

۱۴.۸ با استفاده از مطالب گفته‌شده نتیجه می‌شود $f'(0) = -2$

۱۵.۸ $dy/dx = ۳xy - ۲y$

۱۶.۸ $x'(t) = ۱۲t^۳ + ۲t$ و $y'(t) = ۱۲t - ۱$.

۲۰.۸ $y = ۲x - ۶$

۲۱.۸ با استفاده از تعریف مشتق تابع وارون، می‌توان نوشت $(f^{-۱})'(۱۱) = -۳$

۲۲.۸ $۲y^۲ = y''y - ۳y'^۲$

۲۳.۸ $y^۴ + y''y - ۳y'^۲ = ۰$.

فصل ۲

۱.۸ $۶x$

۲.۸ با استفاده از مطالب گفته‌شده و نیز تعریف مشتق نتیجه می‌شود ۲

۳.۸ با مشتق‌گیری داریم $f'(۲) = ۶(۲) = ۱۲$

۴.۸ تابع f در همه جا مشتق‌پذیر است.

۵.۸ -۶. نزول می‌کند.

Calculus and Analytic Geometry

By

Vahid Damanafshan

Faculty Member Of The Kermanshah University

The Kermanshah University Press

2019