محاسبه پذیری و منطق

مدرس:

دکتر مرتضی منیری

دانشگاه شهید بهشتی

کارگاه تابستانی نظریه علوم کامپیوتر

پژوهشگاه دانش های بنیادی

تابستان ۱۳۹۱

مقدمه:

شروع نظریه محاسبه‌پذیری به دهه ۱۹۳۰ میلادی برمی‌گردد، یعنی زمانی که کامپیوترها ساخته نشده بودند. این زمانی است که منطق‌دانان بزرگی چون چرچ، گودل، کلینی، پُست و تورینگ، در تلاش برای پاسخگویی به برخی سؤالات در حوزه منطق و مبانی ریاضی، از قبیل وجود یا عدم وجود یک دستگاه اصل موضوعی مناسب برای ریاضیات، تلاش کردند تعاریفی دقیق و ریاضی‌وار برای مفاهیمی نظیر الگوریتم و محاسبه بیابند.

هدف اصلی بنیان گزاران منطق ریاضی، ارائه مدل ریاضی برای استنتاج ها به منظور بررسی دقیقتر آنها بوده است. ایده اولیه این کار به ریاضیدان و فیلسوف بزرگ آلمانی قرن هفدهم، لایب نیتز، بر می گردد. اما قدم های اصلی را جرج بول حدود دویست سال بعد برداشت. بول روش های جبری و نمادی را در مطالعه منطق به کاربرد. قدم های اصلی دیگر در پی ریزی منطق ریاضی را افرادی چون گوتلب فرگه، جوزپه پئانو و برتراند راسل در اوخر قرن نوزدهم و اوایل قرن بیستم برداشتند.

مبحث کلیدی در منطق، بررسی استنتاج ها از جهت معتبر بودن و یا معتبر نبودن است

یک استنتاج، برای مثال در منطق گزاره ها، چه زمانی معتبر است؟ یک شیوه برای تعریف دقیق اعتبار یک استنتاج در منطق گزاره ها آن است که آن را معتبر نامیم، هر گاه با فرض راستی مقدمات آن، نتیجه آن نیز راست باشد. این رویکرد، یک رویکرد اصطلاحا معنایی است و بر حسب ارزش گزاره ها تعریف می شود. روش دیگر آن است که یک فرآیند صوری اثبات را معرفی کنیم و یک استنتاج را هنگامی معتبر گوییم که به کمک مقدمات

استنتاج، بتوان نتیجه را ثابت کرد این رویکردی اصطلاحا نحوی است و متکی به قواعد نمادی از پیش تعریف شده است.

نظریه مدل ها و نظریه برهان، دو بخش اصلی منطق ریاضی اند که به ترتیب با رویکرد معنایی و نحوی فوق مرتبط اند. قضایای صحت و تمامیت در منطق، معادل بودن دو روش فوق را تضمین می کنند. تعریف اعتبار استنتاج ها در منطق محمولات نیز با دو روش فوق، ولی با پیچیدگی بیشتر امکان پذیر است. در اینجا اشاره می کنیم که اعتبار استنتاج ب ذکر شده در بالا، علیرغم سادگی آن، در منطق گزاره ها قابل اثبات نیست. برای انجام این کار استفاده از منطق محمولات ضروری است.

آیا فرآیند استنتاج را می توان الگوریتمی کرد؟ یعنی آیا الگوریتمی موجود است که اعتبار استنتاج ها را بیازماید. این سئوال زمینه ساز وجود یک شاخه اصلی دیگر منطق ریاضی، به نام نظریه محاسبه پذیری، شده است. هر یک از دو بخش منطق گزاره ها و منطق محمولات را می توان از این دیدگاه مطالعه کرد. در واقع منطق دانان برای حل سئوالاتی از این قبیل، به تجزیه و تحلیل مفهوم الگوریتمی و ارائه تعریف دقیق آن پرداختند. دستاورد غیر منتظره این امر بسیار ارزشمند بود و آن ارائه توصیف نظری کامپیوتر، بسیار پیش از ساخت نمونه های واقعی آن بود. ماشین تورینگ یکی از مشهورترین این توصیفات است که بوسیله آلن تورینگ، منطق دان مشهور، در سال ۱۹۳۶ ارائه شد.

در این زمینه، مختصرا می توان گفت که منطق گزاره ها تصمیم پذیر است، یعنی الگوریتمی موجود است که اعتبار استنتاج های آن را می آزماید. برای مثال، جدول ارزش چنین الگوریتمی را فراهم می کند (گزاره A یک نتیجه منطقی گزاره های $A_1,...,A_n$ است هر گاه $(A_1 \wedge .... \wedge A_n) \rightarrow A$ یک راستگو باشد). ولی، منطق محمولات تصمیم پذیر نیست. البته این پایان داستان نیست. هر چند که منطق محمولات تصمیم پذیر نیست، ولی بخش

هایی از آن چنین اند، برای مثال اگر آن را به محمولات یک متغیره محدود کنیم، حاصل تصمیم پذیر است.از طرف دیگر، الگوریتمی که بر پایه جدول ارزش، اعتبار استنتاج های گزاره ای را می آزماید، الگوریتم موثری نیست.

دلیل این موضوع آن است که این الگوریتم اصطلاحا در زمان نمایی کار می کند و با افزایش تعداد گزاره های اتمی موجود در استنتاج، زمان لازم برای کار آن به طور نجومی افزایش می یابد. نکته مهم دیگر در مورد منطق گزاره های آن است که، همانطور که استفن کوک، یکی از بنیان گزاران شاخه ای مهم از علوم نظری کامپیوتر به نام نظریه پیچیدگی محاسبات، در سال ۱۹۷۹ ثابت کرد، رده ای وسیع از مسائل الگوریتمی (موسوم به NPها) به طور موثر به مسئله تشخیص ارضاپذیر بودن گزاره ها تحویل می شوند. به سادگی می توان دید که این مسئله که به SAT موسوم است، در ارتباط مستقیم با مسئله بررسی الگوریتمی استنتاج های گزاره ای است. با توجه به این موضوعات، عجیب نیست که مسئله یافتن الگوریتم های سریع گزاره ای، اکنون به هدفی مشترک برای منطق دانان و علمای علوم کامپیوتر تبدیل شده است.

همین جا باید ذکر کرد که هنوز الگوریتمی سریع (یعنی الگوریتمی که در زمان چند جمله ای کارکند و یا اصطلاحا به P متعلق باشد) برای بررسی مسائل مطرح شده در مورد منطق گزاره ها موجود نیست. در واقع اگر الگوریتمی متعلق به P برای بررسی راستگو بودن گزاره ها موجود باشد آنگاه جواب یکی از مهمترین مسائل نظریه پیچیدگی، یعنی $NP =^{?} CoNP$، مثبت خواهد بود.

۱) توابع محاسبه پذیر

آشنایی با زبان برنامه نویسی S

نمادها

متغیرهای ورودی $X_1 , X_2 , ...$

متغیر خروجی Y

متغیرهای موضعی $Z_1 , Z_2 , ...$

نشان‌ها $A_1 , B_1 , ...$

قرارداد : مقادیر متغیرهای خروجی و موضعی در شروع برنامه دارای مقدارصفر هستند.

دستورالعمل‌ها (instructions)

تعابیر دستورالعمل‌ها	دستورالعمل‌ها
نمو : مقدار متغیر V را به اندازه ۱ افزایش بده	$V \leftarrow V+1$
نزول: اگر مقدار متغیر V صفر است، آنرا تغییر نده، در غیر اینصورت آن را به اندازه ۱ کاهش بده	$V \leftarrow V-1$
شاخه شرطی : اگر مقدار متغیر V صفر نیست، دستورالعمل با نشان L را انجام بده، در غیر اینصورت دستورالعمل بعدی را انجام بده.	$If\ V \neq 0\ GoToL$.

چند مثال از برنامه‌ها در زبان S:

مثال (a).

[A] If $X \neq ۰$ GoTo B

$Z \leftarrow Z + ۱$

If $Z \neq ۰$ GoTo E

[B] $X \leftarrow X - ۱$

$Y \leftarrow Y + ۱$

$Z \leftarrow Z + ۱$

If $Z \neq ۰$ GoTo A

برنامه فوق تابع $f(x) = x$ **را محاسبه می‌کند.**

مثال (b).

در این مثال می‌خواهیم برنامه‌ای بنویسیم که همان تابع $f(x)=x$ را محاسبه کند ولی با یک امتیاز اضافی که در پایان کار مقدار متغیر ورودی حفظ شده باشد.

```
[A]  If X ≠ ۰ GoTo B
     GoTo C
[B]  X ← X − ۱
     Y ← Y + ۱
     Z ← Z + ۱
     GoTo A
```

در اینجا X به Y و Z منتقل می‌شود

```
[C]  If Z ≠ ۰ GoTo D
     GoTo E
[D]  Z ← Z − ۱
     X ← X + ۱
     GoTo C
```

در اینجا مقدار X به مقدار قبلی برمی‌گردد و مقدار Z صفر می‌شود

توجه: برنامه فوق برای ساخت یک ماکرو به شکل $V \leftarrow V'$ به کار می رود.

مثال (c)

در این مثال با استفاده از ماکروی $V \leftarrow V'$ برنامه‌ای برای محاسبه تابع $f(x_۱, x_۲) = x_۱ + x_۲$ می‌نویسیم.

$Y \leftarrow X_۱$
$Z \leftarrow X_۲$
[B] If $Z \neq ۰$ GoTo A
GoTo E
[A] $Z \leftarrow Z - ۱$
$Y \leftarrow Y + ۱$
GoTo B

مثال (d)

برنامه زیر چه تابعی را محاسبه می‌کند؟

$Y \leftarrow X_۱$
$Z \leftarrow X_۲$
[C] If $Z \neq ۰$ GoTo A
GoTo E
[A] If $Y \neq ۰$ GoTo B
GoTo A
[B] $Y \leftarrow Y - ۱$
$Z \leftarrow Z - ۱$
GoTo C

جواب: تابع جزئی $g(x_۱, x_۲) = \begin{cases} x_۱ - x_۲ & x_۱ \geq x_۲ \\ \uparrow & x_۱ < x_۲ \end{cases}$

توجه کنید که برنامه (d) به اینگونه کار می‌کند که به ترتیب یکی از X_1 (Y) و یکی از X_2 (Z) کم می‌کند تا جائیکه Z برابر با صفر شود. اگر Y زودتر صفر شود برنامه در یک حلقه‌ی نامتناهی خواهد افتاد.

چند تعریف در مورد زبان S

به دلایل تکنیکی، دستورالعمل‌های به شکل $V \leftarrow V$ (V یک متغیر) را نیز در S مجاز می‌شماریم.

۱) یک وضعیت δاز یک برنامه P عبارت است از یک لیست از معادلات به شکل $V = m$، جائیکه همه متغیرهای ظاهر شده در برنامه در لیست دقیقاً یکبار ظاهر شوند (متغیرهای دیگری نیز ممکن است باشند، ضمناً این وضعیت ممکن است لزوماً یک وضعیت ممکن نباشد.)

۲) یک توصیف لحظه‌ای (id) از یک برنامه P با طـول n (تعداد دستورالعمل‌های P) عبارت است از یک زوج (i , δ) جائیکه $1 \le i \le n+1$ و δ یک وضعیت P است (به طور شهودی i نشان دهنده آن است که دستورالعمل i – ام قرار است اجرا شود، $i = n+1$ به معنای آن است که با یک وضعیت پایانی مواجه هستیم.)

۳) اگر (i , δ) یک توصیف لحظه‌ای غیرپایانی باشد آنگاه تالی (i , δ) که خود یک توصیف لحظه‌ای است به نحو طبیعی تعریف می‌شود.

٤) یک محاسبه از یک برنامه P عبارت است از یک دنباله $S_1 , S_2 , ... , S_k$ از توصیف‌های لحظه‌ای P بطوری که S_{i+1} تالی S_i است و S_k یک وضعیت پایانی است.

توابع محاسبه‌پذیر

تعریف

فرض کنید $r_1, \dots, r_m$ عدد و P یک برنامه باشد. وضعیت δ از P را به صورت زیر تعریف کنید: $Y = 0$, $X_1 = r_1$, $X_2 = r_2$, ... , $X_m = r_m$ (مابقی متغیرهای P را نیز صفر فرض می‌کنیم.)

توصیف لحظه‌ای $(1, \delta)$ را توصیف لحظه‌ای آغازی می نامیم.

دو حالت موجود است.

۱) یک محاسبه $S_1, \dots, S_k$ از P شروع شده با توصیف لحظه‌ای آغازی S_1 و خاتمه یافته با توصیف لحظه‌ای پایانی S_k موجود است.

در این حالت $\psi_P^{(m)}(r_1, \dots, r_k)$ را برابر مقدار Y در S_k تعریف می‌کنیم.

۲) محاسبه (متناهی) از نوع فوق موجود نیست. در این حالت گوییم $\psi_P^{(m)}(r_1, \dots, r_k)$ تعریف نشده است.

تعریف و نکته

به ازای یک برنامه P دلخواه، $\psi_P^{(m)}(x_1, \ldots, x_m)$ تابعی است که توسط P محاسبه می‌شود. اگر برنامه P دارای n متغیر ورودی باشد و $m < n$ ، مابقی متغیرها همواره صفرفرض می‌شوند. اگر m از n بزرگتر باشد مابقی متغیرها فراموش می شوند.

مثال

فرض کنید P برنامه مثال c باشد.

$$\psi_P^{(2)}(r_1, r_2) = r_1 + r_2$$
$$\psi_P^{(1)}(r_1) = r_1 + 0 = r_1$$
$$\psi_P^{(3)}(r_1, r_2, r_3) = r_1 + r_2$$

تعریف

تابع جزئی $g(x_1, \ldots, x_m)$ را محاسبه‌پذیر جزئی گوئیم هرگاه برنامه P موجود باشد که $\psi_P^{(m)}(r_1, \ldots, r_m) = g(r_1, \ldots, r_m)$ به ازای هر $r_1, \ldots, r_m$ (دو طرف یا هر دو تعریف شده و مساوی‌اند یا هر دو تعریف نشده‌اند.)

اگر g تام باشد و توسط یک برنامه محاسبه شود آن را محاسبه‌پذیر گوییم.

قضیه

(۱) فرض کنید f محاسبه‌پذیر جزئی باشد. در این صورت ماکرو $W \leftarrow f\ (V_1\ ,\ \ldots\ ,\ V_n)$ موجود است.

(۲) فرض کنید P محمولی محاسبه‌پذیر باشد در اینصورت دستوراتی از نوع زیرموجودند:

$\text{If}\ \ P(V_1\ ,\ \ldots\ ,\ V_n)\ \text{GoTo L}$

۲) توابع بازگشتی اولیه

ترکیب

فرض کنید f تابعی k – متغیره و $g_1 , \dots , g_k$ توابعی n– متغیره باشند. در اینصورت گوییم تابع n– متغیره h که به ازای هر $x_1 , \dots , x_n$ ،

$$h(x_1 , \dots , x_n) = f(g_1(x_1 , \dots , x_n), \dots , g_k(x_1 , \dots , x_n))$$

از f و $g_1 , \dots , g_k$ بوسیله ترکیب به دست آمده است.

بازگشت (با پارامتر)

فرض کنید f تابعی n– متغیره و g تابعی $n+2$– متغیره باشد. فرض کنید f و g تام باشند. در اینصورت گوییم تابع h بوسیله بازگشت از f و g بدست آمده است هرگاه

$$\begin{cases} h(x_1 , \dots , x_{n,0}) = f(x_1 , \dots , x_n) \\ h(x_1 , \dots , x_n , t+1) = g(t , h(x_1 , \dots , x_n , t) , x_1 , \dots , x_n) \end{cases}$$

توابع آغازی

توابع تالی $S(x) = x+1$ ، صفر $n(x) = 0$ و تصویری

$$(1 \le i \le n) \quad \bigcup_i^n (x_1 , \dots , x_n) = x_i$$

را توابع آغازی می‌نامیم.

توابع بازگشتی اولیه:

یک تابع را بازگشتی اولیه گوییم هرگاه یک تابع آغازی باشد یا با کاربرد تعدادی متناهی از قواعد ترکیب و بازگشت از توابع آغازی بدست آید.

مثال

تابع جمع $f(x_1 , x_2) = x_1 + x_2$ **یک تابع بازگشتی اولیه است.**

حل:

$$\begin{cases} f(x, 0) = x \\ f(x, y+1) = f(x, y) + 1 \end{cases} \quad \text{یا دقیقتر} \quad \begin{cases} f(x, 0) = \bigcup_1^1 (x) \\ f(x, y+1) = g(y, f(x, y), x) \end{cases}$$

جائیکه $g(x_1 , x_2 , x_3) = S(\bigcup_2^3 (x_1 , x_2 , x_3))$

قضیه

هر تابع بازگشتی اولیه محاسبه پذیر است.

چند مثال دیگر از توابع بازگشتی اولیه:

(a) تابع ضرب $x \cdot y$

$$\begin{cases} x.۰ = ۰ \\ x.(y+۱) = x.y + x \end{cases}$$

$$h(x\ ,۰) = n(x)$$
$$h(x,\ y+۱) = g(y\ , h(x,\ y)\ , x)$$
$$g(x_۱\ , x_۲\ , x_۳) = f(\bigcup_{۲}^{۳}(x_۱\ , x_۲\ , x_۳), \bigcup_{۳}^{۳}(x_۱\ , x_۲\ , x_۳))$$

ع جمع f

(b) تابع فاکتوریل $x!$

$$\begin{cases} ۰! = ۱ \\ (x+۱)! = x\ !\ .\ S(x) \end{cases}$$

(c) تابع توان x^y

$$\begin{cases} x^{۰} = ۱ \\ x^{y+۱} = x^y\ .\ x \end{cases}$$

(d) تابع ماقبل

$$P(x)=\begin{cases} x-1 & x\neq 0 \\ 0 & x=0 \end{cases}$$

$$\begin{cases} P(0)=0 \\ P(t+1)=t \end{cases}$$

(e) تفاضل تغییر یافته $x \dot{-} y$

$$x \dot{-} y=\begin{cases} x-y & x\geq y \\ 0 & x<y \end{cases}$$

$$\begin{cases} x \dot{-} 0=x \\ x \dot{-} (t+1)=P(x \dot{-} t) \end{cases}$$

(f) تابع قدرمطلق $|x-y|$

$$|x-y|=(x \dot{-} y)+(y \dot{-} x)$$

(g) تابع آلفا

$$\alpha(x)=\begin{cases} 1 & x=0 \\ 0 & x\neq 0 \end{cases}$$

$$\alpha(x)=1 \dot{-} x$$

$$\begin{cases} \alpha(0)=1 \\ \alpha(t+1)=0 \end{cases}$$

برخی محمولات بازگشتی اولیه

$$x = y$$

$$d(x , y) = \begin{cases} ۱ & x = y \\ ۰ & x \neq y \end{cases}$$

$$d(x , y) = \alpha(|x - y|)$$

$$x \leq y$$

$$\alpha(x \dot{-} y)$$

قضیه

اگر P و Q دو محمول بازگشتی اولیه باشند آنگاه $\sim P$، $P \vee Q$ و $P \wedge Q$ نیز بازگشتی اولیه هستند.

$$\sim P = \alpha(P)$$

$$P \wedge Q = P . Q$$

$$P \vee Q \Leftrightarrow \sim(\sim P \wedge \sim Q)$$

قضیه (تعاریف چند حالتی)

فرض کنید P, g, h بازگشتی اولیه باشند

$$f(x_۱ , \ldots , x_n) = \begin{cases} g(x_۱ , \ldots , x_n) & P \\ h(x_۱ , \ldots , x_n) & \neg P \end{cases}$$

در اینصورت f هم بازگشتی اولیه است.

قضیه

فرض کنید $f(t, x_1, \dots, x_n)$ بازگشتی اولیه باشد در اینصورت g, h نیز بازگشتی اولیه هستند جائیکه

$$g(y, x_1, \dots, x_n) = \sum_{t=0}^{y} f(t, x_1, \dots, x_n)$$ **و**

$$h(y, x_1, \dots, x_n) = \prod_{t=0}^{y} f(t, x_1, \dots, x_n)$$

سورهای محدود

اگر محمول $P(t, x_1, \dots, x_n)$ بازگشتی اولیه باشد، آنگاه دو تای زیر نیز هستند:

$$\forall t \le y \;\; P(t, x_1, \dots, x_n) \;\; , \;\; \exists t \le y \;\; P(t, x_1, \dots, x_n)$$

اثبات: توجه کنید که

$$\forall t \le y \;\; P(t, x_1, \dots, x_n) \;\; \Leftrightarrow \;\; \prod_{t=0}^{y} P(t, x_1, \dots, x_n)$$

و

$$\exists t \le y \;\; P(t, x_1, \dots, x_n) \;\; \Leftrightarrow \;\; \left[\sum_{t=0}^{y} P(t, x_1, \dots, x_n) \right] \neq 0$$

مثال

۱) عادکردن بازگشتی اولیه است.

$$y|x \Leftrightarrow \exists t \le x \;\; y.t = x$$

۲) اول بودن بازگشتی اولیه است.

$$Prime(x) \Leftrightarrow [x > ۱ \wedge \forall t \le x \, (t = ۱ \vee t = x \vee \neg t|x)]$$

قضیه (کمینه سازی محدود)

اگر $P(t, x_۱, \dots, x_n)$ بازگشتی اولیه باشد، آنگاه تابع زیر

$$f(y, x_۱, \dots, x_n) = \min_{t \le y} P(t, x_۱, \dots, x_n)$$

نیز بازگشتی اولیه است.

۳) P_n n – امین عدد اول (استثنائاً فرض می‌کنیم $P_۰ = ۰$) به عنوان تابعی یک متغیره بازگشتی اولیه است.

نکته: کمینه سازی (نامحدود)

به ازای محمول $P(x_1, \ldots, x_n, y)$، $\min_y P(x_1, \ldots, x_n, y)$ برابر با کوچکترین y تعریف می‌شود که $P(x_1, \ldots, x_n, y)$ برقرار باشد، در غیر اینصورت تعریف نشده فرض می‌شود.

برای مثال، $x - y = \min_z [y + z = x]$ که به ازای $x < y$ تعریف نشده است.

محمول $P(x,y)$ وجود دارد بطوریکه تابع $\min_y P(x, y)$ تام باشد ولی بازگشتی اولیه نباشد.

قضیه

اگر $P(x_1, \ldots, x_n, y)$ یک محمول محاسبه پذیر باشد و

$g(x_1, \ldots, x_n) = \min_y P(x_1, \ldots, x_n, y)$ آنگاه g محاسبه‌پذیر جزئی است.

اثبات: برنامه زیر g را محاسبه می‌کند:

```
[A]  If P (X₁ , ... , Xn , Y) GoToE
     Y ← Y + ۱
     GoTo A
```

۳) کد گذاری و مساله توقف

تعریف:

(x,y) کد $= <x,y> = (2y+1) \dot{-} ۱$ ^{x}۲

قضیه (زوج سازی):

توابع $<x,y>$ و $\ell(z)$ و $r(z)$ دارای خواص زیرند:

۱) بازگشتی اولیه اند

۲) $r(<x,y>)=y \ , \ \ell(<x,y>)=x$

۳) $<\ell(z),r(z)>=z$

٤) $\ell(z),r(z)\le z$

کد گذاری دنباله ها با طول دلخواه : $(a_1,....,a_n)$ کد $=\prod_{i=1}^{n} P_i^{a_i}$

نکته

$[a_1,....,a_n]=[b_1,....,b_n]\Rightarrow a_i=b_i \quad i=1,...,n$

$[a_1,...,a_n]=[a_1,....,a_n,0]$

$[0,a_1,....,a_n]\neq[a_1,...,a_n]$

نکته:

به ازای هر n ثابت، تابع $[a_1,...,a_n]$یک تابع بازگشتی اولیه است.

می خواهیم تابع $(x)_i$را بگونه ای تعریف کنیم که اگر $x=[a_1,...,a_n]$ آنگاه $(x)_i=a_i$. برای این منظور تعریف می کنیم:

تعریف: $$(x)_i=\min_{t\le x}(\sim P_i^{t+1}\mid x)$$

توجه کنید که $(x)_0=0$ **و** $(0)_i=0$.

تعریف:

$$Lt(x)=\min_{i\le x}[(x)_i\neq 0\ \wedge\ \forall j\le x\ \ (j\le i\ \vee\ (x)_j=0)]$$

مثال:

$$x=20=2^2.5^1=[2,0,1]\Rightarrow (x)_3=1,$$

$$(x)_4=(x)_5=...=(x)_{20}=0$$

$$\Rightarrow L_t(20)=3$$

توابع $(x)_i$ **و** $L_t(x)$**، بازگشتی اولیه هستند.**

قضیه:

الف) $$([a_1,\ldots,a_n])_i = \begin{cases} a_i & 1 \le i \le n \\ 0 & \text{در غیر اینصورت} \end{cases}$$

ب) **اگر** $n \ge L_t(x)$ **آنگاه** $[(x)_1,\ldots,(x)_n] = x$

کد گذاری عددی برنامه ها

ترتیب زیر را برای متغیرها و نشان ها در نظر می گیریم:

$$Y, X_1, Z_1, X_2, Z_2, X_3, Z_{3,\ldots}$$

$$A_1, B_1, C_1, D_1, E_1, A_2, B_2, C_2, D_2\ , E_2, A_3, \ldots.$$

تعریف:

مکان نشان L **در لیست فوق** $= \#(L)$

مکان متغیر V **در لیست فوق** $= \#(V)$

تعریف:

فرض کنید I یک دستور (نشان دار یا بی نشان) باشد. تعریف می کنیم:

$$\#(I) = < a, < b, c > >$$

جائیکه:

۱. اگر I بی نشان باشد آنگاه $a = 0$، اگر I دارای نشان L باشد آنگاه $a = \#(L)$

۲. $C = \#(V) - 1$، جائیکه V متغیر ظاهر شده در I است.

۳. اگر I، $V \leftarrow V$ باشد آنگاه $b = 0$،

اگر I، $V \leftarrow V + 1$ باشد آنگاه $b = 1$،

اگر I، $V \leftarrow V - 1$ باشد آنگاه $b = 2$.

٤. اگر I، $IF \;\; V \neq 0 \;\; GoTo \;\; L'$ باشد، آنگاه $b = \#(L') + 2$

مثال:

۱. کد دستور بی نشان $X \leftarrow X + 1$ برابر است با: $< 0, < 1,1 >> = < 0,5 > = 10$

۲. کد دستور فوق با نشان A برابر است با: $< 1, < 1,1 >> = 21$

نکته:

به ازای هر عدد داده شده q، یک دستور العمل منحصر به فرد I موجود است بطوریکه $\#(I) = q$ (بنابراین کد گذاری دستورها، یک به یک است.)

تعریف (کد یک برنامه) :

اگر برنامه P دارای دستورات $I_1,...,I_k$، تعریف می کنیم

$$P \text{ کد } = \#(P)=[\#(I_1),\#(I_2),...\#(I_k)]-1$$

برای جلوگیری از ابهام، فرض می شود که دستور آخر برنامه از نوع $Y \leftarrow Y$ نباشد (با این کار هر عدد برنامه منحصر بفردی را معین می کند.)

مثال:

۱. برنامه ساده

$$[A] \quad X \leftarrow X+1$$

$$IF \quad X \neq 0 \quad GoTo \; A$$

دارای کد $[\#(I_1)=21\ ,\ \#(I_2)=46]-1$است که برابر است با $2^{21}.3^{46}-1$!

می خواهیم برنامه ای پیدا کنیم که کد آن ۱۹۹ باشد، داریم

$$199+1=200=2^3.3^0.5^2=[3,0,2]$$

پس برنامه مورد نظر دارای سه دستور است که دستور دوم آن دستور بی نشان $Y \leftarrow Y$ است. از طرفی داریم:

$$\begin{cases} 2=<0,1>=<0,<1,0>> \\ 3=<2,0>=<2,<0,0>> \end{cases}$$

پس دستور اول $[B] \quad Y \leftarrow Y$ و دستور سوم $Y \leftarrow Y+1$ است.

تعریف (محمول توقف)

برنامه با کد y روی ورودی x سرانجام متوقف می شود $\Leftrightarrow HALT(x,y)$

قضیه:

$HALT(x,y)$ یک محمول محاسبه ناپذیر است.

اثبات: برهان خلف.

فرض کنید محاسبه پذیر باشد. پس محمول یک موضعی $HALT(x,x)$ نیز چنین است.
برنامه Pرا بصورت زیر تعریف کنید:

$$[A] \text{ IF } HALT(X,X) \;\; GoTo \;\; A$$

فرض کنید $\#(P) = y_0$ در اینصورت به ازای هر x داریم :

$$HALT(x,y_0) \Leftrightarrow \sim HALT(x,x)$$

سپس با قرار دادن y_0 به جای x به تناقض زیر می رسیم:

$$HALT(y_0,y_0) \Leftrightarrow \sim HALT(y_0,y_0)$$

پس $HALT(x,y)$ نمی تواند محاسبه پذیر باشد.

فرضیه چرچ که در زیر می آید یک حکم اثبات ناپذیر در مورد الگوریتم ها است. الگوریتم مفهومی ریاضی نیست که اثبات های ریاضی وار در مورد آن بکار آید اما این فرضیه مورد قبول اکثر علمای علوم کامپیوتر است.

فرضیه چرچ: هر تابعی که توسط یک الگوریتم قابل محاسبه باشد توسط یک برنامه در زبان مورد بحث ما محاسبه می شود.

نتیجه :

مسئله توقف حل ناپذیر است. به عبارت دیگر هیچ الگوریتمی موجود نیست که به ازای x و y داده شده در مورد درستی $HALT(x,y)$ تصمیم بگیرد.

٤) معادل بودن محاسبه پذیری و بازگشتی بودن

تابع جهانی

تعریف

به ازای هر $n>0$ تعریف می کنیم،

$$\varphi^{(n)}(x_1,...,x_n,y)=\psi_P^{(n)}(x_1,...,x_n)$$

جائیکه $\#(P)=y$. در حالتی که $n=1$ ، معمولا به جای $\varphi^{(1)}$ ، φ خواهیم نوشت.

قضیه جهانی

به ازای هر $n>0$، تابع $\phi^{(n)}(x_1,...,x_n,y)$ محاسبه پذیر جزئی است.

اثبات:

برای اثبات قضیه، به ازای هر $n>0$، باید یک برنامه (جهانی) U_nساخت که $\phi(n)$را محاسبه کند، یعنی

$$\psi_{U_n}^{(n+1)}(x_1,...,x_{n+1})=\varphi^{(n)}(x_1,...,x_n,x_{n+1})$$

یک راه دیگر توجیه درستی این قضیه آن است که از فرضیه چرچ استفاده کنیم. تابع ϕ^n بوضوح بصورت الگوریتمی قابل محاسبه است. پس برنامه ای وجود دارد که آنرا محاسبه می کند !

تعریف

برنامه با کد y روی ورودی $x_1,...,x_n$ $\Leftrightarrow$ $STP^{(n)}(x_1,...,x_n,y,t)$

بعد از حداکثر t مرحله ، متوقف می شود.

قضیه

به ازای هر $n>0$، محمول $STP^{(n)}(x_1,...,x_n,y,t)$ بازگشتی اولیه است.

قضیه (شکل نرمال)

فرض کنید $f(x_1,...,x_n)$ یک تابع محاسبه پذیر جزئی باشد.

در اینصورت یک محمول بازگشتی اولیه $R(x_1,...,x_n,y)$ موجود است بطوریکه

$$\ast \quad f(x_1,...,x_n)=\ell(\min_z R(x_1,...,x_n,z))$$

اثبات: فرض کنید y_0 کد برنامه ای باشد که f را محاسبه می کند. تعریف کنید:

$$R(x_1,...,x_n,z)\Leftrightarrow[STP^{(n)}(x_1,...,x_n,y_0,r(z))\wedge(r(SNAP^{(n)}(x_1,...,x_n,y_0,r(z)))_1=\ell(z)]$$

یعنی برنامه با کد y_0 در حداکثر $r(z)$ مرحله متوقف می شود (به توصیف لحظه ای پایانی می رسد) و مقدار متغیر y (یعنی خروجی) برابر $\ell(z)$ است. یعنی $\ell(z)=f(x_1,...,x_n)$ اگر سمت راست ❊ تعریف شده باشد.

تعریف

یک تابع را بازگشتی جزئی می گوییم هر گاه از توابع آغازی با کاربرد تعدادی متناهی از قواعد ترکیب، بازگشت و کمینه سازی بدست آمده باشد (اگر یک تابع بازگشتی جزیی تام باشد آنرا بازگشتی می نامیم).

قضیه

محاسبه پذیر جزئی بودن ⇔ بازگشتی جزئی بودن

اثبات

⇒ : از قضایای بخش های قبلی نتیجه می شود توابع آغازی تالی، صفر و تصویری محاسبه پذیرند و عملگرهای سه گانه مزبور محاسبه پذیر جزئی بودن را حفظ می کنند.

⇐ : بنا به قضیه شکل نرمال هر تابع محاسبه پذیر جزئی، بازگشتی جزئی نیز هست (توجه کنید که R بازگشتی است و توابع min و ℓ بازگشتی جزئی بودن را حفظ می کنند).

قضیه

یک تابع محاسبه پذیر است اگر و تنها اگر از توابع آغازی با کاربرد تعدادی متناهی از قواعد ترکیب، بازگشت و کمینه سازی سره (یعنی کمینه سازی با فرض وجود کوچکترین عضو) بدست آمده باشد.

اثبات

⇒ واضح است،

⇐ توجه کنید که در اثبات قضیه شکل نرمال، اگر $\ell(\min_z R(x_1,...,x_n,z))$ تام باشد، $\min_z R(x_1,...,x_n,z)$ نیز چنین بوده است.

۵) مجموعه های بطور بازگشتی شمارا

تعریف

مجموعه $B \subseteq \mathbb{N}$ را بطور بازگشتی شما را (یا r.e.) گوییم هر گاه تابع محاسبه پذیر جزئی $g(x)$ موجود باشد بطوریکه

$$B = \{x \in \mathbb{N} \mid g(x)\downarrow\}$$

نکته

بنا به تعریف فوق، اگر B یک مجموعه r.e. باشد آنگاه برنامه ای موجود است بطوری که روی ورودی x، متوقف می شود هر گاه $x \in B$ و متوقف نمی شود هر گاه $x \notin B$. در واقع به سادگی می توان دید که دو شرط فوق معادلند. ضمنا، مفهوم برنامه را می توان با الگوریتم تعویض کرد (صورتی از تز چرچ)

قضیه

اگر B بازگشتی باشد، آنگاه B r.e. است.

قضیه

بازگشتی $B \Leftrightarrow B$ و $\bar{B}$ r.e.

قضیه

اگر B و C دو مجموعه r.e. باشند، آنگاه $B \cup C$ و $B \cap C$

تعریف

فرض کنید $\varphi(x,y)$ همان تابع جهانی باشد. تعریف می کنیم

$$W_n = \{x \in \mathbb{N} \mid \varphi(x,n)\downarrow\}$$

قضیه (شمارش)

یک مجموعه B r.e. است اگر و تنها اگر $n \in \mathbb{N}$ ای موجود باشد که $B = W_n$

اثبات

توجه کنید که $\phi(x,0),\phi(x,1),...$ شمارشی از همه تابع محاسبه پذیر جزئی یک متغیره است.

نکته و تعریف

بنا به قضیه شمارش، $W_0,W_1,W_2,....$ شمارش از همه مجموعه های r.e. است .

تعریف:

$$K = \{n \in \mathbb{N} \mid n \in W_n\}$$

قضیه: K مجموعه ای r.e. است ولی محاسبه پذیر نیست.

اثبات

چون $\phi(n,n)$ تابع محاسبه پذیر جزئی است (قضیه جهانی) و

$$K = \{n \in \mathbb{N} \mid \varphi(n,n)\downarrow\}$$

پس K r.e. است.

اگر K محاسبه پذیر بود، آنگاه $\bar{K}$ نیز می بایست r.e. باشد، پس i موجود می بود که $\bar{K} = W_i$ در اینصورت به تناقض زیر می رسیدیم: $i \in K \Leftrightarrow i \in W_i \Leftrightarrow i \in \bar{K}$

قضیه

فرض کنید $S \neq \varnothing$. شرایط زیر معادلند:

الف) S r.e. است

ب) S برد یک تابع بازگشتی اولیه است.

ج)S برد یک تابع بازگشتی است.

د)S برد یک تابع بازگشتی جزئی است.

۵) برخی قضایای مشهور در نظریه محاسبه پذیری

قضیه پارامتر $(S-m-n)$

فرض کنید $n,m>0$ داده شده باشند. در این صورت تابع بازگشتی اولیه $S_m^n(u_1,u_2,...,u_n,y)$ موجود است بطوری که

$$\phi^{(m+n)}(x_1,...,x_m,u_1,...,u_n,y)=\phi^{(m)}(x_1,...,x_m,S_m^n(u_1,...,u_n,y)$$

قضیه بازگشت

فرض کنید $g(z,x_1,...,x_m)$ یک تابع محاسبه پذیر جزئی باشد. در اینصورت عدد e موجود است بطوریکه

$$\phi_e^{(m)}(x_1,...,x_m)=g(e,x_1,...,x_m)$$

نتیجه

عدد eموجود است بطوریکه $\phi_e(x)=e$ به ازای هر x.

اثبات: تابع محاسبه پذیر $g(z,x)=U_1^2(z,x)=z$ را در نظر بگیرید. بنا به قضیه بازگشت عدد e موجود است $\phi_e(x)=g(e,x)=e$

قضیه نقطه ثابت

فرض کنید که $f(z)$ یک تابع محاسبه پذیر باشد. در اینصورت عدد e موجود است بگونه ای که $\forall x\ \phi_{f(e)}(x) = \phi_e(x)$

اثبات

فرض کنید $g(z,x) = \phi_{f(z)}(x)$ یک تابع محاسبه پذیر جزئی باشد. بنا به قضیه بازگشت، عدد e موجود است که: $\phi_e(x) = g(e,x) = \phi_{f(e)}(x)$

مثال

۱. عدد e موجود است بطوریکه $\phi_e = \phi_{e^2}$

۲. عدد t موجود است بطوریکه $\phi_t = \phi_{t+1}$

٦) تحویل پذیری چند به یک

تعریف

گوییم مجموعه A به مجموعه B تحویل پذیر چند به یک است ($m-$ تحویل پذیر است) و می نویسیم $A \leq_m B$،

هر گاه تابع محاسبه پذیر $f:\mathbb{N}\to\mathbb{N}$ موجود باشد به قسمیکه

$$x \in A \Leftrightarrow f(x) \in B$$

قضیه

فرض کنید $A \leq_m B$. در اینصورت:

الف) اگر B محاسبه پذیر باشد، آنگاه A محاسبه پذیر است.

ب) اگر B r.e. باشد، آنگاه A r.e. است.

نتیجه

فرض کنید $K_0 = \{x \in \mathbb{N} \mid \varphi_{r(z)}(\ell(x))\downarrow\}$ به عبارت دیگر $K_0 = \{< x,y > \mid \varphi_y(x)\downarrow\}$. در اینصورت K_0 مجموعه ای r.e. است ولی محاسبه پذیر نیست.

اثبات:

از آنجا که K_0 برابر با دامنه تابع محاسبه پذیر جزئی $\varphi^{(2)}(\ell(x), r(x))$ است، r.e. است.

برای آنکه نشان دهیم که K_0 محاسبه پذیر نیست، کافی است نشان دهیم که $K \leq_m K_0$.

برای این منظور تعریف کنید: $f(x) =< x,x >$

تعریف

یک مجموعه A را $m-$ کامل گوییم هر گاه A r.e. باشد و ضمنا به ازای هر مجموعه r.e. B، $B \leq_m A$.

مثال: K_0 یک مجموعه $m-$ کامل است.

حل

$$A = \{x \in \mathbb{N} \mid g(x)\downarrow\} = \{x \in \mathbb{N} \mid \varphi(x, z_0)\downarrow\} = \{x \in \mathbb{N} \mid < x, z_0 > \in K_0\}$$

نکته

۱. اگر $A \leq_m B$ و $B \leq_m C$ آنگاه $A \leq_m C$.

۲. اگر A $m-$ کامل باشد و B r.e. باشد و $A \leq_m B$، آنگاه B نیز $m-$ کامل است.

نکته: به کمک مثال و نکته فوق می توان نشان داد که، K یک مجموعه $m-$ کامل است.

فرض کنید Γ یک کلاس از توابع محاسبه پذیر جزئی یک متغیره باشد. فرض کنید R_Γ مجموعه همه اندیس های اعضای Γ باشد، $R_\Gamma = \{t \mid \phi_t \in \Gamma\}$

قضیه رایس

فرض کنید Γ کلاسی از نوع فوق باشد با این شرط اضافه که $\Gamma \neq \varnothing$ و ضمنا Γ شامل همه توابع محاسبه پذیر جزئی یک متغیره نباشد. در اینصورت R_Γ محاسبه پذیر نیست.

ایده اثبات : K یا $\bar{K}$ به R_Γ ، $m-$ تحویل پذیر است.

۷) اراکل و محاسبه پذیری نسبی

فرض کنید یک جعبه سیاه (اراکل) داده شده باشد که به ازای هر x به ما بگوید که آیا $\phi(x,x)\downarrow$ یا خیر. البته بنابراین تز چرچ، رفتار این اراکل بوسیله الگوریتم ها قابل توصیف نیست.

به طور طبیعی برنامه هایی مطرح می شوند که مجهز به این اراکل هستند. این موضوع را می توان در مورد هر تابع محاسبه ناپذیر دیگر نیز مطرح کرد.

به طور صوری:

فرض کنید G یک تابع جزئی یک متغیره باشد (نه لزوما محاسبه پذیر)

در زبان S، به جای دستورات به فرم $V \leftarrow V$ دستوراتی به فرم $V \leftarrow O(V)$ را اضافه می کنیم که عمل آنها به شکل زیر است:

اگر $V = m$ و $G(m)\downarrow$ آنگاه V مقدار $G(m)$ را می گیرد.

اگر $V = m$ و $G(m)\uparrow$ آنگاه با اجرای این دستور برنامه متوقف می شود.

مفاهیم $\psi^{(m)}_{P,G}(r_1,\ldots,r_m)$، $G-$ محاسبه، تابع $G-$ محاسبه پذیر جزئی و .... به طریق مشابه جلسات قبل تعریف می شوند.

نکته

اگر P یک محمول باشد و A یک مجموعه ، آنگاه مفهوم تابع $P-$ محاسبه پذیر (جزئی) و یا $A-$ محاسبه پذیر (جزئی) به نحو طبیعی تعریف می شود (برحسب تابع مشخصه آنها) .

قضیه جهانی نسبی شده

فرض کنید $\phi_G^{(n)}(x_1,...,x_n,y)$ **همان تابع** $\psi_{P,G}^{(n)}(x_1,...,x_n)$ **باشد، جائیکه** P **یک** $G-$ **برنامه است که کد آن** y **است. در اینصورت اگر** G **تام باشد آنگاه تابع** $\phi_G^{(n)}(x_1,...,x_n,y)$، $G-$ **محاسبه پذیر است.**

تعریف

فرض کنید $G:\mathbb{N}\to\mathbb{N}$ **یک تابع جزئی باشد. در اینصورت تعریف می کنیم**

$STP_G^{(n)}(x_1,...,x_n,y,t)\Leftrightarrow$ ***بعد از*** $x_1,...,x_n$ **با ورودی** y **کد** $G-$ **برنامه** ***حداکثر*** t ***مرحله، متوقف می شود.***

قضیه

اگر G **تام باشد، آنگاه محمول** $STP_G^{(n)}(x_1,...,x_n,y,t)$، $G-$ **محاسبه پذیر است.**

قضیه (قضیه پارامتر نسبی)

فرض کنید $m,n>0$ **داده شده باشند. در اینصورت تابع بازگشتی اولیه** $S_m^n(u_1,....,u_n,y)$ **موجود است بطوری که به ازای هر تابع تام** G،

$$\phi_G^{(m+n)}(x_1,...,x_n,u_1,...,u_n,y)=\phi_G^{(m)}(x_1,...,x_m,S_m^n(u_1,...u_n,y))$$

تعریف (تورینگ تحویل پذیری)

مجموعه A به مجموعه B، تورینگ تحویل پذیر است هر گاه مجموعه A، B – محاسبه پذیر باشد. در این حالت می نویسیم $A \leq_t B$

قضیه

الف) $A \leq_t A$ **به ازای هر مجموعه** A

ب) اگر $A \leq_t B$ **و** $B \leq_t C$ **آنگاه** $A \leq_t C$**، به ازای هر سه مجموعه** A,B,C

قضیه

$$A \leq_m B \Rightarrow A \leq_t B$$

اثبات:

فرض کنید $f : \mathbb{N} \xrightarrow{\text{محاسبه پذیر}} \mathbb{N}$ و $x \in A \Leftrightarrow f(x) \in B$

برنامه زیر A را ، B–محاسبه می کند:

$$X \leftarrow f(X)$$
$$X \leftarrow O(X)$$
$$Y \leftarrow X$$

قضیه

اگر A، $m-$ کامل برای کلاس r.e. ها باشد، آنگاه $\bar{A}$ r.e. نیست (پس A محاسبه پذیر نیست.)

اثبات: فرض کنید $\bar{A}$ r.e. باشد. پس از آنجا که A $m-$ کامل است داریم $\bar{A} \leq_m A$.
حال مجموعه K را که r.e. است ولی $\bar{K}$ r.e. نیست در نظر بگیرید.

داریم $K \leq_m A$ پس $\bar{K} \leq_m \bar{A}$. بنابراین $\bar{K} \leq_m A$. ولی در اینصورت می بایست $\bar{K}$ r.e. باشد که نیست.

تعریف

$$A \oplus B = \{2x : x \in A\} \bigcup \{2x+1 : x \in B\}$$

قضیه

الف) $A \leq_t A \oplus B$، $B \leq_t A \oplus B$

ب) اگر $A \leq_t C$ و $B \leq_t C$، آنگاه $A \oplus B \leq_t C$

تعریف

فرض کنید G یک تابع تام (یک یا چند متغیره) باشد. یک مجموعه B را $G-r.e.$ گوییم هر گاه یک تابع $G-$ محاسبه پذیر جزئی g موجود باشد بطوری که

$$B = \{x \in \mathbb{N} \mid g(x)\downarrow\}$$

تعریف: $W_n^G = \{x \in \mathbb{N} \mid \varphi_G(x,n)\downarrow\}$

قضیه (شمارش نسبی شده)

یک مجموعه B، $G-r.e.$ **است اگر و تنها اگر** n**ای باشد که** $B = W_n^G$

تعریف: $G' = \{n \in \mathbb{N} \mid n \in W_n^G\}$ **جهش یافته** G

مثال: $K = I'$

قضیه

G' **یک مجموعه** $G-r.e.$ **است ولی** $G-$ **محاسبه پذیر نیست.**

اثبات: نسبی شده قضیه نظیر در مورد K

تعریف
$$\begin{cases} G^{(0)} = G \\ G^{(n+1)} = (G^{(n)})' \end{cases}$$

قضیه

$\varnothing^{(n+1)}$ **یک مجموعه** $\varnothing^{(n)}-r.e.$ **است ولی** $\varnothing^{(n)}-$ **محاسبه پذیر نیست. در اینجا منظور از** $\varnothing$**، تابع مشخصه آن است که همواره ارزش صفر دارد.**

۷) تصمیم ناپذیری منطق محمولات مرتبه اول

الفبای زبانهای مرتبه اول

مجموعه الفبای یک زبان مرتبه اول L، شامل موارد زیر است:

همه روابط گزاره ای، نمادهای $\forall$ و $\exists$ (سور عمومی و سور وجودی)، یک مجموعه شمارای $x_1, x_2, \ldots$ (متغیرها) و پرانتزها) و (.

بعلاوه، الفبای L می تواند شامل موارد زیر باشد:

الف) یک مجموعه از نمادهایی چون P که نماد محمولی نامیده می شوند. به هر محمول P ، یک عدد طبیعی n نسبت داده می شود. اگر به P عدد n نسبت داده شده باشد، گوییم P یک محمول $n-$ موضعی است.

ب) یک مجموعه از نمادهایی چون C که نماد ثابت نامیده می شوند.

ج) یک مجموعه از نمادهایی چون f که نماد تابعی نامیده می شوند. به هر نماد تابعی f ، یک عدد طبیعی n نسبت داده می شود. اگر به f عدد n نسبت داده شده باشد، گوییم f یک نماد تابعی $n-$ موضعی است.

متغیرها را گاهی به شکل غیر صوری با $x, y, y_1, y_2, \ldots$ نشان خواهیم داد.

تعریف (ترم های L یا L– ترم ها)

الف) هر متغیر و هر نماد ثابت L یک ترم L است.

ب) اگر $t_1,...,t_n$ ترم های L و f یک نماد تابعی n–موضعی L باشد، آنگاه $f(t_1,...,t_n)$ یک ترم L است.

تعریف (L- فرمولها)

مجموعه فرمولهای L، L- فرمولها، به صورت بازگشتی تعریف می شوند:

الف) هر فرمول اتمی L، یک L– فرمول است، یعنی هر عبارت به شکل $R(t_1,...,t_n)$ جایی که R نماد محمولی n– موضعی و t_i ها ترم اند.

ب) اگر φ و ψ دو L– فرمول باشند آنگاه موارد زیر نیز L– فرمولند:

$$(\neg\varphi)،\ (\varphi\wedge\psi)،\ (\varphi\vee\psi)،\ (\varphi\to\psi)،\ (\varphi\leftrightarrow\psi)،\ ((\forall x_i)\varphi)،\ ((\exists x_i)\varphi).$$

تعریف (رخداد آزاد)

یک رخداد یک متغیر x_i در یک فرمول را آزاد گوییم، هر گاه آن رخداد در دامنه عمل یک سور $(\forall x_i)$ یا $(\exists x_i)$ قرار نداشته باشد، در غیر اینصورت آن را محدود می نامیم. بنا به قرارداد x_i در $\forall x_i$ یا $\exists x_i$ نه محدود است و نه آزاد. یک متغیر در یک فرمول آزاد است هر گاه حداقل یک رخداد آزاد در آن داشته باشد.

تعریف (جمله): یک فرمول مرتبه اول را جمله خوانیم هر گاه فاقد متغیر آزاد باشد.

تعریف (L- ساختار)

منظور از یک L– ساختار M، عبارت است از یک مجموعه ناتهی M (موسوم به دامنه M)، همراه با موارد زیر:

الف) به ازای هر نماد ثابت C از L، یک عنصر مشخص C^M از M (تعبیر C در M)،

ب) به ازای هر نماد تابعی $n-$ موضعی f از L، یک تابع $f^M : M^n \to M$ (تعبیر f در M)،

ج) به ازای هر نماد محمولی $n-$ موضعی P از L، یک رابطه P^M، جائیکه $P^M \subseteq M^n$ (تعبیر P در M).

مثال: زبان حسابی و ساختارحسابی

فرض کنید L دارای پارامترهای f_1 ، f_2 ، f_3 ، R و P باشد، جائیکه f_1 ، f_2 ، f_3 به ترتیب، نمادهای تابعی ۱- موضعی، ۲ – موضعی، و ۳ – موضعی و R و P محمولهای ۲- موضعی هستند. در اینصورت $N = (\mathbb{N}, 0, S, +, ., =, <)$ یک L– ساختار است.

در اینجا، S تابع تالی است: $S(x) = x+1$. این زبان L را زبان حسابی می نامیم و با L_{ar} نشان می دهیم. ضمنا N را ساختار حسابی می نامیم.

توجه کنید که ساختارهای کاملا متفاوتی را می توان برای زبان حسابی در نظر گرفت.(مثال بزنید).

در ساختار حسابی، فرمول $\forall x_1 \exists x_2 P(x_1, x_2)$ به این صورت تعبیر می شود: «به ازای هر عدد طبیعی، عددی طبیعی بزرگتر از آن وجود دارد.» توجه کنید که، سورها به اعضای دامنه ساختار (یعنی مجموعه اعداد طبیعی) اشاره دارند و نه مثلا توابع و یا محمولات تعریف شده روی اعداد طبیعی. پسوند «مرتبه اول» به این موضوع اشاره می کند. اصل خوش ترتیبی که معمولا به صورت زیر بیان می شود یک خاصیت مرتبه اول اعداد طبیعی نیست: هر زیر مجموعه نا تهی از اعداد طبیعی یک کوچکترین عضو دارد.

تعریف (معنا شناسی تارسکی)

یک L– جمله φ را منطقا معتبر نامیم هر گاه به ازای هر L– ساختار M، $M \models \varphi$.

$M \models \varphi$ یعنی با تعبیر کردن نمادهای ثابت، تابعی و محمولی ظاهر شده در φ با ثوابت، توابع و روابط نظیر در M حاصل کار جمله ای درست در ساختار Mاست.

منطقا معتبر بودن در منطق مرتبه اول، نظیر راستگو بودن در منطق گزاره هاست. منطقا معتبر بودن یک فرمول به شکل ظاهری آن بستگی دارد و نه به معنای نمادهای به کار رفته در آن.

مثال: جمله $\exists x_2 \forall x_1 \varphi \to \forall x_1 \exists x_2 \varphi$ منطقا معتبر است ولی $\forall x(\varphi \vee \psi) \to (\forall x \varphi \vee \forall x \psi)$ منطقا معتبر نیست.

تصمیم ناپذیری منطق محمولات مرتبه اول که توسط منطق دان نامی چرچ، ثابت شده است. این بدان معنی است که در حالت کلی برای یک زبان مرتبه اول L، مسئله تشخیص اینکه آیا یک L– جمله داده شده φ، منطقا معتبر است یا نه، به طور الگوریتمی

حل ناپذیر است. برای اثبات این موضوع از $m-$ تحویل پذیری بصورت غیرصوری استفاده می شود.

به عبارت دیگر، یک مسئله معروف در نظریه محاسبه پذیری که می دانیم که تصمیم ناپذیر است را در نظر گرفته و آنرا به مسئله تشخیص جمله های منطقا معتبر تحویل می کنند. یعنی به ازای هر نمونه از مسئله مزبور، به صورت الگوریتمی $L-$ جمله φ را می سازند به طوری که جواب سئوال در مورد نمونه مزبور مثبت باشد اگر و تنها اگر فرمول φ منطقا معتبر باشد. مسئله ای که به مسئله تشخیص اعتبار جمله ها تحویل می شود، مسئله مشهور PCP (مسئله تناظر پست) است.

تعریف (مسئله تناظر پست)

فرض کنید (S_1,t_1) و (S_2,t_2) و ... و (S_k,t_k) یک دنباله از زوج های مرتب باشد بگونه ای که هر یک از S_i ها و t_i ها یک رشته با طول مثبت از ۰ و ۱ هاست. مسئله تناظر پست عبارت از این مسئله است که آیا دنباله ای از اندیس ها چون $i_1,...,i_n$ موجود است به گونه ای که رشته های حاصل از کنار هم گذاری دنباله های $S_{i_1},...S_{i_n}$ و $t_{i_1},...,t_{i_n}$ با هم مساوی باشند؟

مثال: فرض کنید C نمونه زیر از مسئله PCP باشد:

$$C=(1,101),(10,00),(011,11).$$

در این صورت دنباله ۳و ۲ و ۳و ۱ از اندیس ها دارای خاصیت زیر است:

$$S_1S_3S_2S_3=t_1t_3t_2t_3=101110011$$

قضیه: مسئله PCP به طور الگوریتمی حل ناپذیر است. ایده اثبات تحویل مسئله توقف به مسئله PCPاست.

۸) مروری بر قضایای گودل

ساختار حسابی $N = (\mathbb{N}, 0, S, +, ., =, <)$ را در نظر بگیرید. اهمیت این ساختار از دیدگاه ریاضی بر کسی پوشیده نیست. اعداد طبیعی نه تنها اهمیت ذاتی دارند بلکه بسیاری از ساختارهای مهم دیگر ریاضی از قبیل اعداد حقیقی را می توان بر پایه آنها بنیان گذارد.

حال $Th(N)$، مجموعه همه جملات حسابی راست، را در نظر بگیرید. آیا این مجموعه به عنوان یک نظریه، اصل پذیر است؟ البته! مجموعه همه اعضای $Th(N)$ را می توان به عنوان مجموعه اصولی برای خودش در نظر گرفت. اما وقتی که یک ریاضیدان صحبت از اصل پذیر بودن یک نظریه ریاضی می کند، چیزی مانند هندسه اقلیدسی را مد نظر دارد که با تعدادی متناهی اصول مشخص و ملموس می توان قضایای بسیار زیادی را در هندسه ثابت کرد. یا بطور کلی تر، مجموعه اصول موضوعی نظریه مورد نظر می بایست تصمیم پذیر باشد. یعنی به ازای یک جمله داده شده زبان، می بایست بتوان بصورت مکانیکی تحقیق کرد که آیا این جمله یک اصل موضوع نظریه مورد بحث است یا نه؟

آیا $Th(N)$ به این معنی اصل پذیر است؟ کورت گودل، منطق دان نامی، در سال ۱۹۳۵ با اثبات اینکه این سئوال پاسخ منفی دارد، جامعه ریاضیدانان را با شگفتی مواجه کرد. این قضیه نتایج عمیقی در حوزه فلسفه ذهن و هوش مصنوعی دارد. برخی از این قضایا این نتیجه را گرفته اند که کامپیوتر ها هیچگاه جانشین ریاضیدانان نخواهد شد.

در واقع گودل نشان داد که هیچ زیر مجموعه تصمیم پذیری از $Th(N)$ نمی تواند همه اعضای آن را نتیجه دهد. اثباتهای گودل در نهایت پیچیدگی هستند و آوردن آنها با همه جزئیات در اینجا ناممکن است. تنها به آوردن شمایی از آنها بسنده خواهیم کرد. یک شگرد بسیار کار آمد که گودل به کار برد، شگرد کد گذاری فرمولهای حسابی به وسیله اعداد طبیعی است. (شبیه آن چیزی که ما در مورد برنامه ها بکار بردیم).

فرض کنید T یک نظریه حسابی معقول باشد (یعنی تصمیم پذیر باشد و توانایی اثبات خواص پایه ای اعداد طبیعی را داشته باشد). به عبارت دیگر، فرض کنید که T یک کاندیدای احتمالی برای اصل پذیر کردن $Th(N)$ باشد. گودل با نسبت دادن اعداد طبیعی به نمادهای زبان و کد گذاری دنباله های متناهی بر حسب کد اعضای آنها به نحوی خاص، فرمولها، دنباله های فرمولها، برهانهای T ودر نهایت قضیه های T را کد گذاری کرد. این کد گذاری به گونه ای بود که به ازای یک n داده شده ، امکان بررسی اینکه «آیا n کد یک برهان در T است یا نه؟»، یا اینکه، «آیا n کد یک قضیه T است یا نه؟» موجود بود. سپس گودل مطالبی را ثابت کرد که او را در ردیف بنیان گزاران نظریه محاسبه پذیری قرارداد. او ثابت کرد که از الگوریتمی بودن بررسی های فوق می توان نتیجه گرفت که فرمولهای حسابی چون $Proof(x)$ و $Th(x)$ موجودند به قسمی که به ازای هر n، T ثابت میکند $proof(n)$اگر و تنها اگر n کد یک برهان در T باشد، و T ثابت میکند$Th(n)$ اگرو تنها اگر n کد یک قضیه T باشد، به عبارت دیگر، «برهان بودن» و «قضیه بودن» در T قابل بیان هستند. کد یک فرمول ψ را معمولا عدد گودل ψ می نامند. آنرا با $\overline{\psi}$ نشان خواهیم داد.

سپس گودل حقیقت مهم زیر را ثابت کرد.

قضیه (لم قطری) : فرض کنید $\varphi(x)$ فرمولی حسابی باشد. در این صورت جمله ای حسابی چون δ موجود است بطوریکه T ثابت کند $\delta \leftrightarrow \varphi(\bar{\delta})$.

به کمک این لم، قضیه اول ناتمامیت گودل نتیجه می شود.

قضیه (قضیه اول ناتمامیت گودل) : فرض کنید T مانند فوق باشد. فرض کنید جمله δ به گونه ای باشد که T ثابت کند $\delta \leftrightarrow \neg Th(\bar{\delta})$

در این صورت T دو جمله δ , $\neg\delta$ را ثابت نمی کند.

توجه: در قضیه فوق، یکی از δ یا $\neg\delta$ می بایست در ساختار حسابی راست باشد. پس جمله ای راست موجود است که T نمی تواند آنرا ثابت کند.

قضیه ای دیگر که گودل ثابت کرد، قضیه دوم ناتمامیت گودل، مثالی مشخص از یک جمله حسابی راست که T توانایی اثبات آن را ندارد ارائه می کند. فرض کنید $Con(T)$ جمله $\neg Th(0=1)$ باشد، یعنی جمله ای که بیان می کند « T سازگار است».

قضیه (قضیه دوم ناتمامیت گودل)

$Con(T)$ در T اثبات نمی شود.