حکمت و فلسفه

سال چهارم، شماره اول، بهار ۱۳۸۷، صص ۲۹ ـ ۴۹

Hekmat va Falsafeh
(Wisdom and Philosophy)
Vol. 4, No. 1, May 2008

تصمیم‌پذیری سیستم‌های هوشمند

سید محمدعلی حجتی*

مرتضی مزگی‌نژاد**

چکیده

الگوسازی از ذهن و ارائهٔ مدلی که قابلیت‌های پیچیدهٔ ذهن را داشته باشد یکی از افق‌های توانمندی بشر است. اگرچه عمدهٔ تلاش‌ها در این زمینه بیش‌تر از نیم قرن سابقه ندارد و با دستاوردهای مسحورکنندهٔ خود یکی از پدیده‌های شگفتی‌ساز شده است، اما هر چه این رویا رنگ واقعیت بیش‌تری به‌خود می‌گیرد مشکلات بزرگ‌تری را بر سر راه نظریه‌پردازان هوش مصنوعی قرار می‌دهد. دو مسئلهٔ عمده‌ای که در این زمینه در مقاله حاضر بررسی خواهد شد عبارت‌اند از: الف ـ آیا سیستم‌های هوشمند قادر خواهند بود هر مسئله‌ای را حل کنند؟؛ ب ـ آیا می‌توان رابطه‌ای میان این مسئله و تصمیم‌ناپذیری منطق محمولات مرتبهٔ اول برقرار کرد؟؛ که هریک از آنها در درون خود شامل مسائل جزئی‌تری هستند که به‌طورخلاصه عبارت‌اند از: ۱ ـ چه سیستمی را می‌توان سیستم هوشمند نامید؟ و ۲ ـ نحوهٔ حل مسئله در سیستم هوشمند به چه صورتی است؟ و ۳ ـ چه سیستمی را تصمیم‌پذیر گویند؟

فرضیه‌های مطرح‌شده در این مقاله نیز بدین‌قرارند:

---

*. استادیار گروه فلسفه دانشگاه تربیت مدرس:
تهران، بزرگراه جلال آل احمد، پل نصر، دانشگاه تربیت مدرس، دانشکده علوم انسانی، گروه فلسفه.
hojatima@modares.ac.ir

**. مربی دانشگاه بیرجند.
mezginegad@gmail.com

الف -- سیستم‌های هوشمند از رویه‌ای الگوریتمی تبعیت می‌کنند. اگر بتوان مسئله‌ای یافت که الگوریتم‌پذیر نباشد، می‌توان نتیجه گرفت آن مسئله برای سیستم هوشمند حل‌ناپذیر است. ب - حل مسائل در هر سیستم هوشمندی متأثر از منطق حاکم بر آن است؛ بنابراین، عدم حل برخی از مسائل توسط سیستم منعکس‌کنندهٔ ناتوانی منطق (محمولات) در ارائهٔ الگوریتمی متناهی برای برخی از فرمول‌هاست تا مشخص کند آیا آن فرمول‌ها معتبرند یا خیر.

واژگان کلیدی: *سیستم‌های هوشمند، مسئلهٔ توقف، ماشین تورینگ، تصمیم‌ناپذیری، منطق محمولات تبیین.*

* * *

مقدمه

آغاز تحقیقات در زمینهٔ هوش مصنوعی (Artificial Intelligen) ـ که در این مقاله به جای تکرار آن از عبارت AI استفاده می‌کنیم ـ به‌صورت رسمی به اوایل قرن بیستم برمی‌گردد. اولین کارها توسط وارن مک کلود و والتر پیتز انجام شد. آنها در کارهای خود، علاوه بر بررسی نحوهٔ عملکرد مغز انسان، از تحلیل رسمی منطق گزاره‌ها متعلق به راسل و وایتهد بهره جستند (راسل،۱۳۸۴، ص۴).

از همان آغاز دیدگاه‌های متفاوتی دربارهٔ هوش و ادراک مطرح شد که از آن جمله می‌توان به فیزیکالیسم[۱] (physicalism)، کارگردگرایی[۲] (functionalism)، و نظریهٔ سازگاری[۳] (coherence theory) اشاره کرد. در سال ۱۹۵۰م، آلن تورینگ عملاً به‌جای ارائهٔ تعریفی از هوشمندی، آزمونی را طراحی کرد که به کمک آن می‌توان هوشمندی ماشین را سنجید. او در مقالهٔ مشهور « computing machinery and intelligence» آزمون خود را که بعدها به «تست تورینگ» شهرت یافت ارائه کرد.

تست تورینگ: «فرض کنید شما در یک سمت دیوار، پرده، یا مانعی قرار دارید و به‌صورت «تله تایپ» با آن سوی دیوار ارتباط برقرار می‌کنید و شخصی از آن سوی دیوار نیز از این طریق با شما در تماس است. به‌طبع، مکالمه‌ای بین شما و شخص آن سوی دیوار می‌تواند صورت پذیرد. حال اگر پس از پایان این مکالمه به شما، گفته شود که آن سوی دیوار نه یک شخص که کاملاً از هویت شخص آن سوی دیوار بی‌خبرید، که یک ماشین بوده که پاسخ شما را می‌داده است، آن ماشین ماشینی هوشمند خواهد بود. در غیر این‌صورت، یعنی در صورتی که شما در حین مکالمه به تصنعی‌بودن پاسخ‌ها پی ببرید، ماشین آن سوی دیوار هوشمند نیست و موفق به گذراندن تست تورینگ نشده است.» هرچند تاکنون تلاش‌هایی در جهت پیاده‌سازی تست تورینگ صورت گرفته[۴] اما هنوز هیچ ماشینی موفق به گذر از چنین تستی نشده است. آنچه بعدها برای تورینگ اهمیت پیدا کرد مفهوم الگوریتم بود، زیرا هر سیستم هوشمندی برای انجام هر عملی ناگزیر از یک رویهٔ الگوریتمی تبعیت می‌کند.

الگوریتم

الگوریتم (Booles, 1974, p.48) مجموعه‌ای متناهی از دستورالعمل‌ها برای انجام عملی است که با داشتن حالتی اولیه به حالت پایانی مشخص و متناظری خواهد رسید. الگوریتم‌ها، به‌دلیل پردازش اطلاعات، اهمیتی اساسی و حیاتی دارند، زیرا هوشمندی یک سیستم اساساً متأثر از الگوریتمی است که به سیستم می‌گوید برای انجام یک عمل خاص، مانند محاسبهٔ حقوق کارمندان یا چاپ برگهٔ گزارش دانش‌آموزان، چه مراحل خاصی را با چه نظم خاصی اجرا کند. در این مقاله به **مبنای هوشمندی**، که در واقع داشتن الگوریتمی ساده برای حل مسائل است، بسنده می‌کنیم و خود را درگیر تعاریف متفاوتی که برای هوش شده نمی‌کنیم . الگوریتم‌های مختلف ممکن است یک عمل را با دستورات مختلف در مدت زمان و با تلاش کم‌تر یا بیش‌تری نسبت به بقیه انجام دهند. آنچه از ویژگی‌های عمدهٔ الگوریتم به‌حساب می‌آید عبارت است از:

۱ . هر الگوریتم دارای مراحل خاصی است که هر مرحله در زمانی محدود و معین انجام می‌شود.

۲ . تعداد مراحل باید معین و محدود باشد.

۳ - یک الگوریتم باید بتواند تکرارپذیر باشد؛ یعنی اگر مراحل آن‌را در رابطه با همان ورودی اعمال کردیم، نتیجهٔ پیشین تکرار شود.

نبود دقت ریاضی درتعریف الگوریتم مشکلاتی را پدید آورد (همیلتن،۱۳۸۳، ص۱۹۲)؛ به همین جهت برای ارائهٔ توصیف دقیقی از این مفهوم، تورینگ الگوریتم را معادل برنامه‌ای ساده قرار داد که اکنون به عنوان ماشین تورینگ شناخته می‌شود (Mendelson, 1997,p.306) . این ابتکار تورینگ زمینه را برای تحقیقات عمیق‌تر فراهم ساخت (در ادامهٔ مقاله به‌روشنی با این برنامهٔ سادهٔ تورینگ آشنا می‌شویم). این تحقیقات در نهایت زمینه را برای آلونزو چرچ (Alvanzo church) فراهم ساخت تا فرضیهٔ خود را ارائه دهد[5]: «*هر آنچه بتوان برای آن الگوریتمی ارائه داد قابل محاسبه با ماشین تورینگ است*» .(Melvin,1987, p.155)

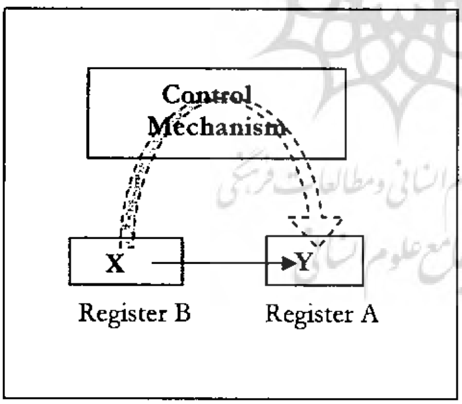
با پیشرفت‌های بسیاری که در علوم کامپیوتر و محاسبه‌پذیری (Theory of Computation) به عمل آمده، میلیون‌ها الگوریتم مورد مطالعه و تحقیق قرار گرفته است. با وجود این، هیچ کس نتوانسته پروسه‌ای را توسعه دهد که الگوریتمیک باشد اما قابل اجرا توسط ماشین تورینگ نباشد. امروزه توافق عمومی بر سر این مسئله هست که سیستم‌های مبتنی بر الگوریتم دقیقاً همان سیستم‌های هوشمندند.

در علوم نظری کامپیوتر، این ماشین سادهٔ تورینگ ماشینی انتزاعی (Abstract Machine) است که دارای قدرت محاسباتی بالاست. ماشین انتزاعی مدلی نظری از سیستم هوشمند است که در ریاضیات و علوم کامپیوتر کاربرد دارد. از این ماشین‌های انتزاعی در تحلیل الگوریتم‌های پیچیده یا شبیه‌سازی اندیشه استفاده می‌شود که بر خلاف سیستم واقعی محدودیتی در منابعی مانند زمان، حافظه و ... ندارند و این امر سبب قدرت محاسبه‌پذیری بالا در آنها می‌شود. به‌عبارت ساده‌تر، هر چند با پیشرفت دانش سیستم‌ها هوشمندتر و قوی‌تر می‌شوند، در عین حال این ماشین انتزاعی ساده قابلیت حل مسائلی را دارد

که آن سیستم‌های پیشرفته قادر به حل آنها هستند (Absoluteastronomy, 2006).

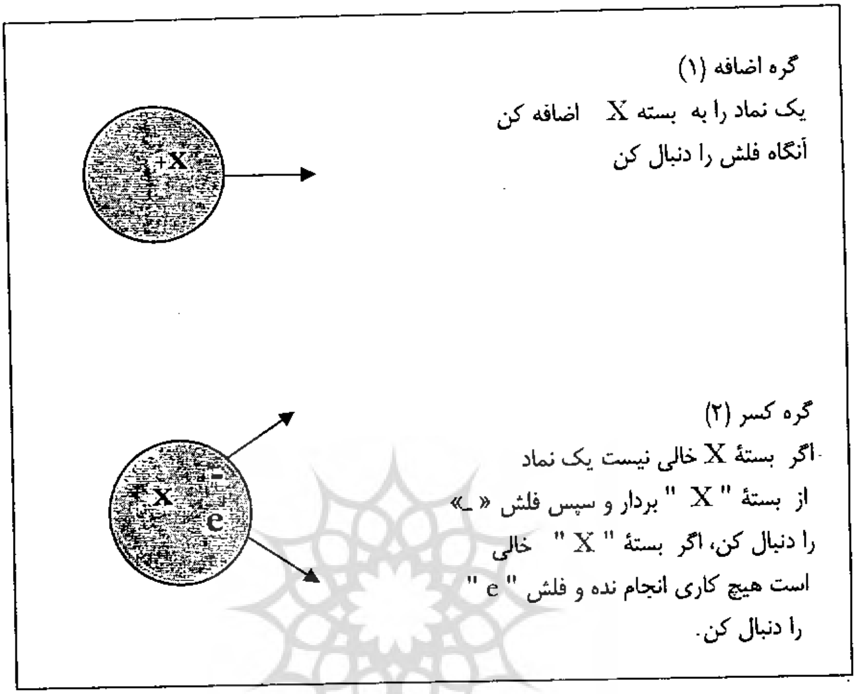
در اواخر قرن بیستم مروین مینسکی (Marvin Minsky) از دانشمندان AI، که نویسندهٔ چندین اثر در زمینهٔ هوش مصنوعی و فلسفه است، الگویی دقیق‌تر از این ماشین انتزاعی ارائه داد[۶] که برای پاسخ دادن به اولین مسئلهٔ مطرح شده مبنی براینکه «آیا سیستم‌های هوشمند قادر خواهند بود هر مسئله‌ای را حل کنند؟» می‌بایست ابتدا نحوهٔ عمل این ماشین انتزاعی را به اختصار بررسی کنیم.

این ماشین که از این به بعد با عبارت RM به آن اشاره می‌کنیم، متشکل از مجموعه‌ای از بسته‌ها (Register) به همراه برنامهٔ کنترلی برای حرکت‌دادن نمادها (counters) از درون این بسته‌هاست (*Stanford Encyclopedia*, 2007, p.2). برنامهٔ کنترل، ماشین را بگونه‌ای هدایت می‌کند که در یک زمان، نمادی را از یک بسته خارج و داخل بستهٔ دیگر کند. به عنوان نمونه، فرض کنید RM می‌خواهد دو عدد را با یکدیگر جمع کند. برای این‌کار ما به دو بسته نیازمندیم. بستهٔ "A" و بستهٔ "B" را به‌گونه‌ای فرض می‌کنیم که نماد "X" در بستهٔ "A" و نماد "Y" در بستهٔ "B" باشد. به ماشین دستور می‌دهیم تا برداشتن نماد از بستهٔ "B" را آغاز کند و هر نمادی را که از B برمی‌دارد در بستهٔ "A" قرار دهد. این پروسه زمانی که بستهٔ "B " کاملاً خالی شد، تمام می‌شود.



شکل (۱) : مدلی از نحوهٔ کار ماشین

برای نمایش دقیق مراحل در این مثال از شکل‌های زیر کمک می‌گیریم. توضیح هر شکل جلوی آن آمده است.



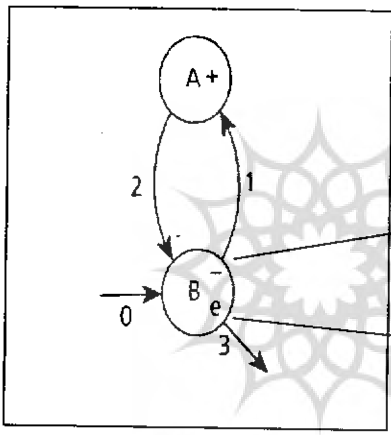
گره‌های ماشین ریجستری (John Nolt ,1997, p.270)

در هر مرحله‌ای این دو عمل می‌تواند تکرار شود. تکرار مراحل وابسته به خالی بودن بسته‌ها در گره شمارهٔ ۲ است. اگر در این مرحله هنوز بسته حاوی نماد باشد، مراحل تکرار می‌شود. با توجه به اینکه RM می‌تواند به‌عنوان ترکیبی از اعمال جمع و کسر نوشته شود، ابتدا گره‌های اضافه و کسر را توضیح می‌دهیم. گره اضافه شامل علامت مثبت و یک فلش است که از آن خارج می‌شود. این فلش پس از اینکه نمادی به بسته اضافه شد، جهت مرحلهٔ بعدی را نشان می‌دهد. جهت فلش از یک گره به گره دیگر نشان‌دهندهٔ ترتیب اعمال این برنامه است. هر گره کم‌کننده (Subtraction node) دارای دو فلش است که یک فلش به‌وسیلهٔ علامت منفی "-" و دیگری توسط "e" مشخص شده است. در صورتی که پس از حذف یک علامت در بسته هنوز علامتی در آن وجود داشته باشد، ماشین در جهت فلش با علامت "-" حرکت می‌کند و اگر پس از حذف یک علامت بسته کاملاً خالی شده باشد، فلش با علامت " e "جهت حرکت را نشان می‌دهد. ممکن است که هر کدام از این فلش‌ها به گره دیگری راهنمایی کند یا به هیچ گره‌ای اشاره نکند و در آنجا برنامه متوقف شود. اکنون دوباره به مثال قبلی برمی‌گردیم که از ماشین خواسته شد نمادهای درون بسته‌ها را با یکدیگر جمع کند. برنامه‌ای که ماشین دنبال می‌کند عبارت است از:

۱ - یک علامت را از بستهٔ " B " بردار.
۲ - این علامت را به بستهٔ "A " اضافه کن.
۳- علامت دیگری را از بستهٔ " B "بردار.
۴ - دوباره این علامت را به بستهٔ "A "اضافه کن.
۵ - مراحل را مانند بالا انجام بده تا بستهٔ " B " خالی شود، آن‌گاه بایست.

با توجه به توضیحات بالا این مراحل را به‌صورت زیر نشان می‌دهیم:

مراحل حرکت ماشین



یک نماد از B کم و به A اضافه می‌کند،

در صورتی که هیچ نماد دیگری در B وجود نداشته باشد.

توضیح شکل بالا

این تصویر شامل دو گره (node) A و B است که گره A جمع‌کننده است و گره B کم‌کننده. فلش‌ها برای سادگی کار شماره‌گذاری شده‌اند. فلش آغازین (entry arrow) با شمارهٔ صفر مشخص شده است، این فلش همیشه باید به‌وسیلهٔ شماره صفر مشخص شود زیرا برنامه برای شروع ابتدا دنبال فلشی با شمارهٔ صفر می‌گردد. شماره‌گذاری دیگر فلش‌ها اختیاری است. فلش (۳)، فلش خروجی نامیده می‌شود. فلش پایان، نقطهٔ توقف یک برنامه است اما همهٔ برنامه‌ها ریجستری دارای نقطهٔ پایان نیستند.

بیان برنامه در قالب منطق (logical programming notation)
(Nolt John ,1997, p.270)

دستورات برنامه را می‌توان در قالب منطقی بیان کرد. ضرورت این امر هنگام بررسی مسئلهٔ دوم مقاله روشن خواهد شد. در اینجا صرفاً این امر را برای بالا بردن دقت دستورات انجام خواهیم داد. در

نمادسازی منطقی از محمول (n+2) موضعی استفاده می‌شود که n تعداد بسته‌های RM است. در برنامهٔ قبل از دو بسته استفاده کردیم، بنابراین در اینجا به محمول چهارموضعی (four places predicate) نیاز داریم. برای مثال، روش خواندن یک حرف محمولی به‌ترتیب زیر است:

R_{twxy} یعنی در زمان "t" ماشین در جهت فلش w حرکت می‌کند، در حالی که دارای "x" نماد در بستهٔ "A" و" y " نماد در بستهٔ B است.

با استفاده از چنین محمولی می‌توانیم رفتار ماشین را در زمان‌های داده شده توصیف کنیم. موضع اول t همیشه بیان‌کنندهٔ زمان و موضع دوم w بیان‌کنندهٔ شماره فلش است، موضع‌های دیگر (x,y,...) نشان‌دهندهٔ تعداد نماد در بسته‌های ماشین هستند. بسته‌ها نیز به ترتیب حروف الفبای لاتین مشخص می‌شوند. در مثال قبل موضع سوم x بیان‌کنندهٔ تعداد نمادها در بستهٔ "A" و موضع چهارم y بیان‌کنندهٔ تعداد نمادها در بستهٔ " B " است.

برای مثال:

$$R\ 3127$$

می‌گوید: در زمان "۳" ماشین در جهت فلش"۱" حرکت می‌کند همراه با دو نماد در بستهٔ "A" و هفت نماد در بستهٔ "B". برنامه توسط روابطی از جملات شرطی نوشته می‌شود، برای مثال: اگر ماشین در زمان "t" در فلان مکان باشد، آنگاه در زمان 't می‌بایست در مکان بعدی باشد. بنابراین:

$$t'3x0 \rightarrow R\ t2x0\ R$$

یعنی اگر ماشین در زمان"t" در جهت فلش "۲" باشد با "X" نماد در بستهٔ "A" و"0" نماد در بستهٔ "B" آنگاه در زمان t' ماشین در جهت فلش"۳" حرکت کند.

برای اضافه یا کم کردن نمادی توسط ماشین می‌توانیم از متغیرها با علامت (′) به معنای تالی استفاده کنیم. مثلاً:

$$R\ t1xy \rightarrow R\ t'2x'y$$

این دستور می‌گوید: اگر در زمان t ماشین در جهت فلش "۱" است همراه با x نماد در بستهٔ A و y نماد در بستهٔ B، آنگاه در زمان 't باید در جهت فلش ۲ حرکت کند؛ در حالی که 'x نماد در بستهٔ A و y نماد در بستهٔ B وجود دارد. پس گره اول در فلوچارت قبلی، چنین دستوری را به ماشین می‌دهد که در جهت فلش"۱"حرکت و نمادی را به بستهٔ A اضافه کند و سپس در جهت فلش"۲" پیش رود. تمام

دستورات مثال قبل را می‌توان به این صورت نوشت:

$$R\ t0xy' \rightarrow R\ t'1xy$$
$$Rt0x0 \rightarrow Rt'3x0$$
$$R\ t1xy \rightarrow Rt'2x'y$$
$$Rt2xy' \rightarrow Rt'1xy$$
$$Rt2x0 \rightarrow Rt'3x0$$

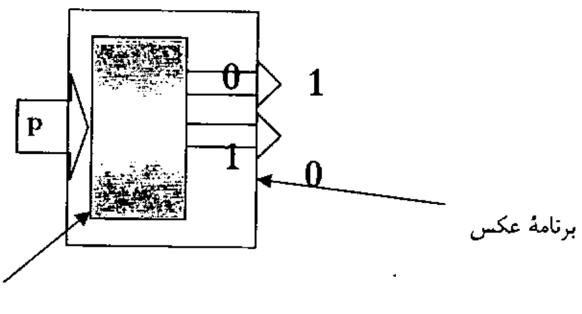
با این توضیحات، نحوهٔ کار RM روشن شد. همچنین چون RM با اعداد سروکار دارد هر داده‌ای که به ماشین بدهیم ابتدا باید به‌صورت عددی کدگذاری شود، به‌گونه‌ای که هر دادهٔ ورودی (item of input) با عددی جایگزین شود[۷]. با این توضیح روشن می‌شود که عملاً نقش برنامهٔ انتزاعی تورینگ توصیف دقیق الگوریتم است. حال با این مقدمهٔ نسبتاً بلند اولین مسئله را بررسی می‌کنیم:

همان‌گونه که در فرضیهٔ اول بیان شد، تمام سیستم هوشمند از رویه‌ای الگوریتمی تبعیت می‌کند. اگر بتوان مسئله‌ای یافت که الگوریتم‌پذیر نباشد، می‌توان نتیجه گرفت که آن مسئله برای هر سیستم هوشمند دیگری حل‌ناپذیر است.

مسئلهٔ توقف (Halting problem)

(Labor, 2006)

اکنون این پرسش را مطرح می‌کنیم: آیا الگوریتمی (برنامه‌ای) وجود دارد که برای هر برنامه‌ای بتواند تشخیص دهد که آیا آن برنامه پس از طی تعداد مراحل متناهی، متوقف خواهد شد یا نه؟ در این بخش نشان خواهیم داد که هر ایده‌ای برای پاسخ به این پرسش، نقض‌کنندهٔ خود (self- contradictor) خواهد بود. فرض می‌کنیم برنامه‌ای وجود داشته باشد که بتواند این مسئله را حل کند؛ یعنی برنامه‌ای که بتواند دیگر برنامه‌ها را تست کند و بگوید آیا آنها متوقف خواهند شد یا خیر. اگر برنامه این قابلیت را داشته باشد، با کمک چنین برنامه‌ای، برنامهٔ عکس توقف (reverse halting program) را می‌توانیم بسازیم.



تعریف: برنامهٔ عکس توقف هر RM، برنامه‌ای است که هرگاه برنامه‌ای مانند p را دریافت کند متوقف می‌شود(0)، اگر و تنها اگر برنامهٔ اصلی با همان داده‌های ورودی متوقف نشود(1)؛ و برعکس، برنامهٔ عکس توقف متوقف نمی‌شود(1)، اگر برنامهٔ اصلی یا همان داده‌های ورودی در ریجسترA متوقف شود(0)..برنامهٔ عکس توقف را به اختصار با RRM نشان می‌دهیم (Nolt John, 1997, p.292).

حال خود برنامهٔ RRM را می‌توان به‌صورت اعداد کُدشده درآورد. RRM هنگامی اعداد کدشدهٔ خود را به‌عنوان داده‌های ورودی دریافت کند چه اتفاقی می‌افتد؟ به‌جای اینکه مستقیماً به این مسئله بپردازیم، ابتدا مثالی شبیه آن می‌آوریم.

پارادکس سرتراش برتراند راسل

(Danziger, 2006, p.2)

مرد آرایشگری را در نظر بگیرید که در روستایی زندگی می‌کند. او سر هر مردی را در روستا اصلاح می‌کند، اگر و تنها اگر خود آن فرد نتواند سر خودش را اصلاح کند. در ظاهر، این امر کاملاً ممکن به‌نظر می‌رسد. هر مرد روستایی یا خودش سرش را اصلاح می‌کند یا آرایشگر، اما هر دو با هم جمع نمی‌شود. اشکال زمانی پیش می‌آید که از خود آرایشگر بپرسیم؛ زیرا سر هر مرد روستایی، از جمله خود آرایشگر را، خود آرایشگر اصلاح می‌کند، اگر و تنها اگر آرایشگر سرخود را اصلاح نکند. و این تناقض است که نشان می‌دهد هر ایده‌ای از چنین آرایشگری ناسازگار است و چنین آرایشگری وجود ندارد.

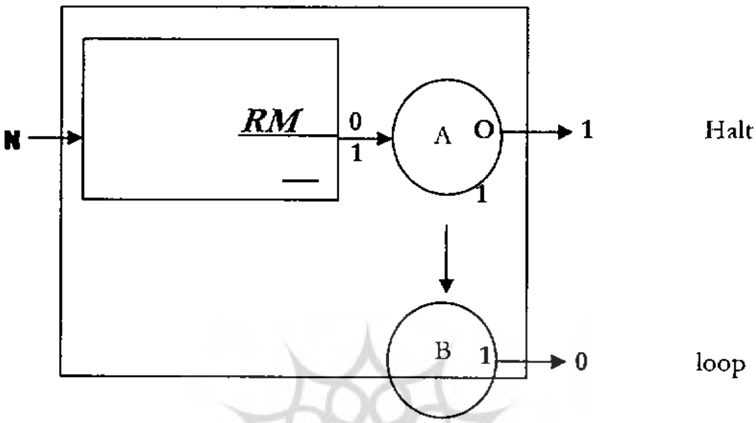
RRM شبیه پارادوکس راسل است، اما در اینجا وظیفهٔ آرایشگر اصلاح است و وظیفهٔ برنامه RRM پاسخ به اعداد کدشده. پاسخ این برنامه، زمانی که اعداد کدشدهٔ خود را دریافت کند، برخلاف زمانی که کدهای دیگر برنامه‌ها را دریافت می‌کند دارای تناقض خواهد بود. با استفاده از دو برهان زیر می‌توان این مسئله را اثبات کرد:

........برای هیچ کدگذاری RRM وجود ندارد (Lewis, 2005, p.4).

اثبات: فرض می‌کنیم برای برخی از برنامه‌های کدشده RRM وجود دارد. پس وقتی برنامه‌ای مانند p را به‌عنوان ورودی دریافت کند RRM متوقف می‌شود، اگر و تنها اگر ورودی p متوقف نشود، و برعکس، اگر برنامهٔ ورودی p متوقف شود برنامهٔ عکس متوقف نمی‌شود. RRM خودش یک برنامه است و می‌توان آن‌را کدگذاری و به‌عنوان ورودی لحاظ کرد. RRM متوقف می‌شود، اگر و تنها اگر برنامهٔ ورودی (یعنی خود RRM) متوقف نشود، و این تناقض است. بنابراین، برای هیچ کدگذاری برنامهٔ عکس توقف وجود ندارد.

.... اگرRM وجود داشته باشد که بتواند مسئلهٔ توقف را حل کند، آنگاه RRM برای برخی کدگذاری‌ها وجود دارد (Lewis,2005, p.4).

برهان: فرض کنید نوعی RM مسئلهٔ توقف را حل می‌کند. حال با کمک RM می‌توانیم به روشنی برنامهٔ ساده RRM را با اضافه کردن دو گره به RM بسازیم، به‌گونه‌ای که اگر خروجی RM صفر باشد آن برنامه متوقف شود و اگر خروجی RM یک باشد آن برنامه همچنان به پیش رود.



THE PROGRAM RRM

چون RM هنگامی که برنامه‌های کدشده را به‌عنوان ورودی دریافت می‌کند متوقف می‌شود و خروجی آن یا 1 است یا 0، پس باید یک فلش خروجی داشته باشد. برای ساخت RRM فلش را به سمت اولین گره ضمیمه می‌بریم. بر طبق این فلوچارت هرگاه برنامهٔ کدشدهٔ n را به عنوان ورودی به بستهٔ A بدهیم، RRM می‌ایستد، اگر و تنها اگر برنامهٔ کدشدهٔ n با داده‌های ورودی n در بستهٔ A نایستد. RRM برنامهٔ عکس توقف خواهد بود. بنابراین، اگر RM وجود داشته باشد که مسئلهٔ توقف را حل کند، آن‌گاه برنامهٔ RRM نیز وجود دارد.

از این دو برهان با کمک قاعدهٔ رفع تالی نتیجه می‌شود RMای که بتواند مسئلهٔ توقف را حل کند، وجود ندارد:

- اگر RMای وجود داشته باشد که مسئلهٔ توقف را حل کند، آن‌گاه RRMای برای برخی از کدگذاری‌ها وجود دارد.
- برای هیچ کدگذاری برنامهٔ RRM وجود ندارد.

....

∴ RMای که بتواند مسئلهٔ توقف را حل کند وجود ندارد.

RM مدلی نظری از سیستمی هوشمند است که دارای قدرت محاسباتی بالاست و بر خلاف سیستم واقعی، محدودیتی در منابعی مانند زمان، حافظه و ... ندارد. همچنین همان‌طور که گفته شد، با پیشرفت دانش هر چند سیستم‌های هوشمندتر و قوی‌تر می‌شوند، باز هم این ماشین انتزاعی ساده قابلیت حل

مسائلی را دارد که آن سیستم‌های پیشرفته قادر به حل آنها هستند. پس چون مسئلهٔ مورد نظر توسط ماشین انتزاعی غیر قابل حل است، بنابراین مسئله‌ای وجود دارد که هیچ سیستم هوشمندی قادر به حل آن نخواهد بود. به عبارت دقیق‌تر، سیستم‌های هوشمند از رویه‌ای الگوریتمی تبعیت می‌کنند، چون این مسئله الگوریتم‌پذیر نیست بنابراین برای سیستم هوشمند حل‌ناپذیر است.

تصمیم‌ناپذیری منطق محمولات

(Church, 1956, p.246)

همان‌طور که در مقدمه گفته شد، وارن مک کلود و والتر پیتز در کارهای خود از تحلیل‌های صوری منطق بهره جستند. یکی از اهداف استفاده از منطق، دستیابی به یک سیستم استنتاجی سازگار برای حل مسائل است. منطق را مانند یک زبان برنامه‌نویسی برای یک ماشین در نظر بگیرید. همان‌طور که یک زبان برنامه‌نویسی به طور معمول از دو قسمتِ تعریف داده‌ها و کد برنامه (دستورات) تشکیل شده است، منطق نیز متشکل از یک زبان و قواعد استنتاج است که این‌دو در کنار یکدیگر سیستم منطقی را به‌وجود می‌آورند. سیستم‌های منطقی متفاوتی ارائه شده که هر کدام دارای قابلیت‌های متفاوتی هستند. منطق محمولاتِ مرتبهٔ اول نیز یکی از این سیستم‌های منطقی است که قواعد و اصول موضوعه خاص خود را دارد. این منطق می‌تواند ساختار استنتاجی یک ماشین هوشمند باشد و سیستم را وادار به پیروی از قواعد استنتاجی خود کند. آنچه داری اهمیت است تأثیر مسئلهٔ توقف (مسئلهٔ اول مقاله) بر این منطق است. منطق محمولاتِ مرتبهٔ اول، به تبع این مسئله، اصطلاحاً تصمیم‌ناپذیر می‌شود. منطق‌دانان یک سیستم منطقی را در صورتی تصمیم‌پذیر گویند که شرایط زیر را داشته باشد:

۱) دستورالعمل‌های ساخت فرمول‌ها و قواعد استنتاجی آن دقیق و متناهی باشد.
۲) این دستورالعمل‌ها کاملاً محاسباتی (الگوریتمی) باشد.
۳) کاربرد این دستورالعمل‌ها نتایج قطعی، و نه تقریبی، به‌دست دهد.
۴) با اعمال این دستورالعمل‌ها بر روی یک فرمول بتوان پس از مراحل متناهی مشخص کرد آیا آن فرمول قضیه‌ای از سیستم هست یا نه (نبوی، ۱۳۷۷، ص۷۸).

اما ساختار منطق محمولات مرتبهٔ اول به‌گونه‌ای است که می‌توان فرمولی را یافت که هیچ دستورالعملی وجود نداشته باشد تا بتواند مشخص کند این فرمول قضیه‌ای از سیستم هست یا خیر. با کمک برهان زیر و آنچه قبلاً گفته شد می‌توان این مسئله را اثبات کرد:

.... اگر منطق محمولات تصمیم‌پذیر باشد، آنگاه برنامهٔ ریجستری وجود دارد که می‌تواند مسئلهٔ توقف را حل کند (Nolt John, 1997, p.269).

برهان: فرض کنید منطق محمولات تصمیم‌پذیر باشد، به این معنی که الگوریتم متناهی برای هر سلسله‌ای از فرمول‌ها وجود دارد که معین می‌کند آیا آن سلسله معتبر است یا نه. حال با کمک فرضیهٔ

چرچ (هر الگوریتمی می‌تواند به‌وسیلهٔ برخی از ماشین‌های انتزاعی نشان داده شود)، می‌توان نتیجه گرفت RMای وجود دارد که با تعداد مراحل متناهی مشخص می‌کند هر سلسله‌ای از فرمول‌ها معتبر است یا نه. می‌توان این مسئله را به این صورت نشان داد:

- اگر فرمول معتبر باشد، RM با نشان دادن 1 می‌ایستد.
- اگر فرمول نامعتبر باشد، RM با نشان دادن 0 می‌ایستد.

از طرفی ماشین انتزاعی و ورودی آن را می‌توان در قالب منطق محمولات بیان کرد، به‌گونه‌ای که ماشین انتزاعی با ورودی مشخص متوقف شود، اگر و تنها اگر زنجیرهٔ مرتبط با آن در منطق محمولات معتبر باشد[۸].

تبدیل RM به فرمول مرتبط با آن در منطق محمولات (Fitting ,1987, p.156):

برای مشخص کردن فرمول مرتبط با یک برنامه با r بسته و ورودی n، ابتدا دو مقدمه را بیان می‌کنیم:

i. یک جمله با فرم $R\,00i0...0$ می‌گوید در زمان 0 ماشین در جهت فلش 0 بوده است، با i نماد در بسته A و تمام دیگر بسته‌ها خالی بوده‌اند.
ii. تمامی دستورات برنامه با متغیرهایشان عموماً مسوّر (quantified) به سور می‌شوند.

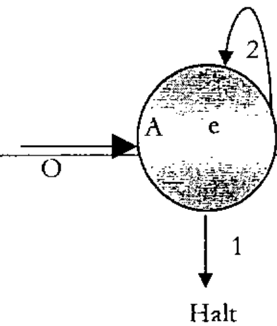
و بالاخره زنجیرهٔ فرمول‌ها به‌صورت زیر مشخص می‌شوند:

۱- اگر یک یا چند فلش وجود داشته باشد، ساختار هر فلش موجود جمله‌ای به شکل زیر خواهد بود:

$$\exists t \exists x_1 \ldots \exists x_r R\, tex_1 \ldots x_r$$

که e شمارهٔ فلش، t متغیری که نشان‌دهندهٔ زمان و $\exists$ علامت سور وجودی است ($\exists a$، یعنی a وجود دارد)، و x...xr متغیرهایی است که نشان‌دهندهٔ تعداد نمادهای هر بسته هستند. این جمله می‌گوید ماشین در زمان معین t با ورودی معین به فلش e می‌رسد، در حالی که دارای x_r نماد است. ساختار دیگر دستورات نیز به‌همین صورت است.

۲- چنین زنجیره‌ای در صورتی معتبر خواهد بود که برنامه با آن ورودی نهایتاً متوقف شود (Booles, 1974, p.52). به‌عنوان نمونه، برنامهٔ زیر با تعداد پنج نماد در ورودی به این صورت نمایش داده می‌شود:



(حالت ماشین هنگام آغاز برنامه)

005 R

1. $\forall t\,(Rt00 \rightarrow Rt'20)$
2. $\forall t \forall x\,(Rt0x' \rightarrow Rt'1x)$
3. $\forall t\,(Rt20 \rightarrow Rt'20)$

$\therefore\ \exists t \exists x\, Rt1x$

(Halt)

این برنامه با دادهٔ ورودی ۵ (تعداد نمادها) متوقف می‌شود. (برنامه فقط یک عدد را کم می‌کند و می‌ایستد). بنابراین، زنجیرهٔ مرتبط با آن معتبر است. (این نتیجه از مقدمات اول و سوم به‌دست می‌آید.) اگر در این مثال به‌جای R005 مقدمهٔ اول را به‌صورت R000 به‌کار بریم، آن‌گاه زنجیره معتبر نخواهد بود. زیرا با ورودی 0 برنامه متوقف نمی‌شود. دستورات این زنجیره برای هر لحظه، عملیات برنامه را توصیف می‌کنند.

حالت ماشین هنگام آغاز برنامه" R005 " است، R005 کار ماشین را در زمان 0 توصیف می کند؛ یعنی ماشین در زمان صفر در جهت فلش 0 حرکت می‌کند در حالی که 5 نماد در بستهٔ آن وجود دارد. این حالت با محمول $Rt0x'$ منطبق است. بنابراین، با استفاده از مقدمهٔ دوم و پس از کم کردن یک شماره از 5، ماشین در موقعیت $Rt'1x$ قرار می‌گیرد که در این مورد $x = 4$ است. چون برای این موقعیت دستور دیگری در مقدمات وجود ندارد، ماشین متوقف می‌شود.

اگر مقدمهٔ اول به‌جای" R 005"، " R 000 " باشد، آنگاه این مقدمه بی‌نهایت دستور را تهیه می‌کند؛ زیرا برنامه پس از گرفتن دادهٔ "R000"، چون عدد بسته صفر است، بنا به دستور ۱ وارد فلش ۲ می‌شود بدون اینکه به این بسته چیزی اضافه کند. به‌عبارت دیگر، در زمان ۱ در جهت فلش ۲ با صفر

عدد در بسته است: R120. برنامه بنا به دستور سوم دوباره بدون اینکه عددی را به بسته اضافه کند در جهت فلش ۲ حرکت می‌کند و این فرایند به‌همین صورت ادامه می‌یابد بی‌آنکه متوقف شود.

1-Rooo	حالت ماشین هنگام آغاز برنامه
2-R120	بنابر حالت ۱ و دستور اول
3- R220	بنا بر حالت ۲ و دستور سوم
4- R320	بنا بر حالت ۳ و دستور سوم
.	
.	
.	

به‌طور کلی اگر t نمایندهٔ زمان، e نمایندهٔ شمارهٔ فلش و cr ......c1 نمایندهٔ شماره‌های داخل بسته‌ها باشد، می‌توانیم محمول Rtec1 ....cr را به این صورت توصیف کنیم: این زنجیره مرتبط است با برنامهٔ P با ورودی i در زمان t، اگر و تنها اگر زمانی که برنامهٔ P با ورودی i شروع به‌کار می‌کند، در زمان t ماشین درجهت فلش e با نمادهای cr ......c1 در بستهٔ مربوط به خود خواهد بود. وظیفهٔ دیگر اثبات پیش‌فرض زیر است که در اثبات برهان به‌کار رفته.

لم: اگر برنامه‌ای با داده‌های ورودی " i " متوقف شود، آنگاه زنجیرهٔ فرمول مرتبط با آن برنامه معتبر است (Nolt 1997, p.273):

برهان: فرض کنید برنامهٔ P با ورودی i نهایتاً متوقف شود. بنابراین، برنامهٔ P در زمانی مانند S و در جهت فلش e با برخی اعداد (نمادهای) Cr ......C1 در بسته‌هایش متوقف می‌شود؛ پس در زمانی قبل ازS نمی‌ایستد. از طرفی برای هر زمانی مانند t اگر برنامهٔ قبل از t متوقف نشود، مقدمات زنجیرهٔ مرتبط با برنامه توصیف معتبری از زمان t را تهیه می‌کنند[۹]. بنابراین، با استفاده از این لم مقدمات زنجیرهٔ مرتبط با برنامه را تهیه می‌کنیم.

Rse C1...... Cr

که از این زنجیره به طور معتبری می‌توان نتیجه گرفت:

$$\exists t \exists y_1 \ldots \exists y_r \, R\, t e y_1 \ldots y_r$$

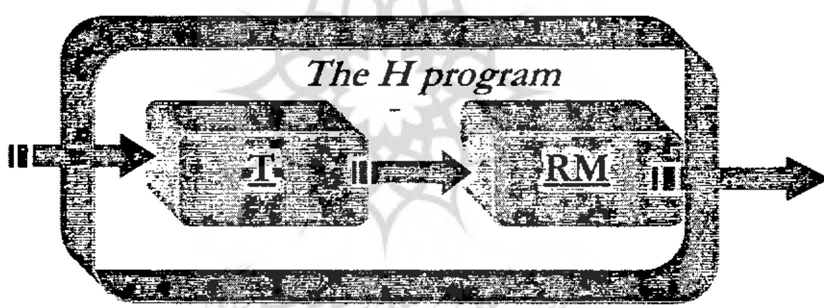
چون این زنجیرهٔ مرتبط از همان مقدمات معتبر به‌دست آمده، این زنجیره نیز معتبر است.

بنابراین، اگر برنامه‌ای با داده‌های ورودی" i " متوقف شود، آنگاه زنجیرهٔ فرمول مرتبط با آن برنامه معتبر است.

با استفاده از این نتیجه، می‌توانیم برنامهٔ مجزای T (Translator) را فرض کنیم (Nolt John, 1997, p.276)، به‌صورتی که ورودی آن برنامه‌هایی مانند P باشد. پس از اینکه برنامهٔ T این ورودی را دریافت کرد، دو عملیات روی آنها انجام می‌دهد.

- نمایشی از زنجیرهٔ فرمول‌های مرتبط را برای P می‌سازد.
- نمایشی از زنجیرهٔ فرمول‌ها را کدگذاری می‌کند و این کد را به‌عنوان خروجی ارائه می‌دهد.

RM را به T ضمیمه می‌کنیم. بنابراین، خروجی T مستقیماً ورودی RM خواهد بود و نتیجهٔ برنامهٔ جدید برنامهٔ H است. اگر بتوان الگوریتمی متناهی برای هر سلسله از فرمول‌ها یافت به‌گونه‌ای که مشخص کند آن سلسله معتبر است یا خیر، خواهیم داشت:



H program : (شکل : ۶)

- H با نشان دادن 1 می‌ایستد، اگر برنامهٔ عددگذاری‌شدهٔ p نهایتاً متوقف شود.
- H با نشان دادن 0 می‌ایستد، اگر برنامهٔ عددگذاری‌شدهٔ p هرگز متوقف نشود.

با این مقدمات می‌توان نتیجه گرفت که H می‌تواند مسئلهٔ توقف را حل کند. بنابراین، اگر منطق محمولات تصمیم‌پذیر باشد، آنگاه برنامهٔ ریجستری وجود دارد که می‌تواند مسئلهٔ توقف را حل کند. از طرفی برنامهٔ ریجستری که بتواند مسئلهٔ توقف را حل کند وجود ندارد، و بنا بر قاعدهٔ رفع تالی می‌توان نتیجه گرفت که منطق محمولات تصمیم‌ناپذیر است که این امر متأثر از مسئلهٔ توقف است که در این مقاله به آن پرداختیم.

نتیجه‌گیری

با مشخص شدن مبنای هوشمندی که در واقع داشتن الگوریتمی ساده برای حل مسائل است و ارائهٔ تعریفی از ماشین انتزاعی مینسکی(RM)، توانستیم با استفاده از دو فراقضیهٔ زیر پاسخ اولین مسئله را به شکل زیر بیان کنیم:

- اگر RMای وجود داشته باشد که مسئلهٔ توقف را حل کند، آنگاه RRM برای برخی از کدگذاری‌ها وجود دارد.
- برای هیچ کدگذاری‌ای برنامهٔ RRM وجود ندارد.

....

∴ RMای که بتواند مسئلهٔ توقف را حل کند وجود ندارد.

RM مدلی نظری از سیستم هوشمند است که دارای قدرت محاسباتی بالاست و چون مسئلهٔ مورد نظر غیر قابل حل است، بنابراین مسئله‌ای وجود دارد که هیچ سیستم هوشمندی قادر به حل آن نخواهد بود. در ادامه با توصیفی از مفهوم تصمیم‌پذیری و برقرار کردن تناظری میان منطق محمولات مرتبهٔ اول و قواعد استنتاجی RM، فراقضیهٔ زیر اثبات شد:

"اگر منطق محمولات تصمیم‌پذیر باشد، آنگاه برنامهٔ ریجستری وجود دارد که می‌تواند مسئله توقف را حل کند."

از طرفی برنامهٔ ریجستری که بتواند مسئلهٔ توقف را حل کند وجود ندارد، بنابراین تصمیم‌ناپذیری منطق محمولات به تبع تصمیم‌ناپذیری سیستم‌های هوشمند اثبات می‌شود.

پی‌نوشت‌ها

۱. فیزیکالیسم یا ماتریالیسم دیدگاهی است که در آن هوش و ادراک را حاصل عملکرد سیستم فیزیکی متشکل از نرون‌ها، سلول‌ها و ساختارهای پشتیبانی‌کنندهٔ آنها می‌داند.
۲. آنچه در این دیدگاه دارای اهمیت است، خواص خروجی / ورودی نرون‌ها است و نه خصوصیت فیزیکی آنها. کارکردگرایی منجر به این عقیده می‌شود که سیستم‌های AI با ساختار مناسب می‌توانند دارای هوش واقعی باشند.
۳. نظریهٔ سازگاری، ساختار درونی یک عامل را در صورتی معقول می‌داند که: اولاً در درون این ساختار گزاره‌ای پذیرفته شود که دلیل کافی بر صدق آن وجود داشته باشد، ثانیاً در درون ساختار هیچ‌گونه تناقضی وجود نداشته باشد و ثالثاً این ساختار نقش کلیدی در انتخاب فعالیت‌های عامل برعهده داشته باشد.
۴. مانند برنامهٔ Eliza یا Artificial Intelligence Markup Languag : (A.I.M.L) ، زبانی برای نوشتن برنامه‌هایی که قادر به chat کردن اتوماتیک باشند.
۵. به همین دلیل برخی این فرضیه را فرضیهٔ چرچ/تورینگ می‌نامند.

۶. ماشین مینسکی با عناوین مختلفی شناخته می‌شود مانند: ماشین محاسبه‌گر (Counter Machine)، زیرا توسط وی معرفی شده وتوسعه پیدا کرده است، یا ماشین برنامه (Progtam Machine) و معروف‌تر از همه ماشین ریجستری (Register Machine).

۷. الگوریتمی هم برای کدگذاری (encoding) و هم برای کدخوانی (decoding) هر زنجیره‌ای از فرمول‌ها وجود دارد. این امری شدنی است؛ در حقیقت شیوه‌های زیادی برای این کار وجود دارد. کدگذاری را از RM که به زبان منطقی درآمده آغاز می‌کنیم، چنین برنامه‌هایی صرفاً زنجیره‌ای از نمادها (Sequence of symbols) هستند. این مفهوم منطقی در حقیقت متشکل از ده علامت ابتدایی است:

$$\rightarrow \ . \ 0 \ t \ x \ y \ z \ R \ ' \ *$$

با نماد 0 آشنا هستید، t , x , y, z نیز متغیر (Variable) هستند. فلش نشان‌دهندهٔ شرط (Condition) است، R حرف محمولی است و علامت (') علامت تالی است. (*) به نمادها اضافه شده است تا زمانی که به متغیرهای بیش‌تری نیاز است متغیر جدید بسازد. (بنابراین متغیر x با x^* ، x^{**} و x^{***} متفاوت است.) آخرین علامت، نقطه است که نشان‌دهندهٔ به پایان‌رسیدن یک فرمول است. اعداد متناسب با هر علامت نوشته می‌شوند. حال اگر بخواهیم برنامه‌ای که در این نوشته به آن پرداخته شده است را کدگذاری کنیم، ابتدا می‌بایست به هر علامتی رقمی بین ۰ تا ۹ بدهیم. فرض کنید این کار را بدین صورت انجام دادیم:

.	t	x	y	z	→	R	*	'	0
0	1	2	3	4	5	6	7	8	9

در این صورت، مثلاً دستورالعمل $Rt'1xy \rightarrow Rt0xy'$ را می‌توان با چنین کدی نشان داد (عدد ۱ را به‌صورت تالی صفر محسوب می‌کنیم): 61923856189823

۸. مرتضی مزگی‌نژاد، (۱۳۸۴)، تصمیم‌ناپذیری منطق محمولات مرتبهٔ اول؛ پایان‌نامهٔ کارشناسی ارشد، تهران: دانشگاه تربیت مدرس.

۹. لم: برای هر زمان t، اگر برنامه قبل از زمان t متوقف نشود، مقدمات زنجیرهٔ مرتبط با آن برنامه تعبیر صادقی از زمان t را ارائه می‌کنند.

اثبات: این لم را می‌توان با استقراء ریاضی (mathematical induction) اثبات کرد.

قبل از ارائهٔ اثبات چند نکته را تذکر می‌دهیم، اولاً صدق یک فرمول در اینجا به معنای مطابقت با مقدمهٔ برنامهٔ متناظر است. ثانیاً فرمول‌های مطابق با برنامه تعبیر صادقی از زمان t ارائه می‌کنند، اگر مطابق مقدمات برنامه در زمان t باشند.

گام بنیادین (Basis case): باید نشان داد اگر برنامه‌ای قبل از زمان 0 متوقف نشود، مقدمات زنجیرهٔ مرتبط با آن توصیف صادقی از زمان صفر ارائه می‌دهند. اما چون مقدمهٔ اول توصیف صادقی از زمان صفر است این امری بدیهی است. (در واقع همان فرض شروع برنامه است.)

گام استقرا (Inductive Step): با استفاده از مقدمهٔ استقرا می‌بایست این شرط را اثبات کنیم که:

اگر هر برنامه‌ای که در زمان t متوقف نشود، مقدمات زنجیرهٔ مرتبط با آن برنامه توصیف صادقی از زمان t ارائه کند، آنگاه اگر یک برنامه قبل از زمان t +1 متوقف نشود، مقدمات زنجیرهٔ مرتبط به‌صورت معتبری توصیف صادقی از زمان t +1 ارائه خواهند کرد.

فرض کنید برنامه‌ای قبل از زمان t +1 متوقف نشود. در این صورت این برنامه در زمان t نیز متوقف نمی‌شود. با استفاده از فرض استقرا می‌توان نتیجه گرفت که مقدمات زنجیرهٔ مرتبط با آن برنامه توصیف صادقی از زمان t را ارائه می‌کنند. حال چون ماشین در زمان t متوقف نشده و الگوریتم به انتها نرسیده است، حرکت در مسیری غیر از فلشِ خروج خواهد بود و ماشین می‌بایست در جهت فلشی غیر از فلش خروج در زمان t قرار داشته باشد؛ این فلش را x می‌نامیم. چون x فلش خروجی نیست باید بخشی از دستورات برنامه مربوط به این فلش باشد، به‌گونه‌ای که به برنامه بگوید زمانی که در جهت فلش x حرکت می‌کند چه عملی انجام دهد. بنابراین، چون مقدمات زنجیره به‌طور معتبری توصیف صادقی از برنامه را در زمان t و t +1 ارائه می‌کنند پس مسئلهٔ مورد نظر اثبات می‌شود. بنابراین، با کمک لم مطرح شده می‌توان نتیجهٔ زیر را گرفت:

برای هر RM و رقم غیر منفی i می‌توان زنجیرهٔ مرتبطی از منطق محمولات را مشخص کرد، به‌گونه‌ای که RM با ورودی i متوقف شود، اگر و تنها اگر زنجیرهٔ مرتبط با i و RM معتبر باشد.

منابع

راسل، (۱۳۸۴). ***هوش مصنوعی***. ترجمهٔ رامین رهنمون. تهران: ناقوس.

موحد، ضیا. (۱۳۶۸). ***درآمدی به منطق جدید***. تهران: سازمان انتشارات و آموزش انقلاب اسلامی.

نبوی، لطف‌الله. (۱۳۷۷). ***مبانی منطق جدید***. تهران: دفتر نشر آثار علمی دانشگاه تربیت مدرس.

همیلتن، آ .گ. (۱۳۸۳). ***منطق برای ریاضی‌دانان***. ترجمهٔ محمدعلی پورعبدالله. تهران: مؤسسهٔ چاپ و انتشارات آستان قدس رضوی.

Absolute Astronomy Encyclopedia. (2004) "Abstract machine", in http://www.absoluteastronomy.com/encyclopedia/a/ab/abstract_machine.htm.

Ackermann, W. (1954). *Solvable Cases of the Decision problem.* Amsterdam: North Holland Publishing Company.

Bernays, P. & Schonfinkel, M. (1928). *Zum Entscheidungsproblem der Mathematischen Logik. Mathematischen Annalen*, vol. 99.

Booles, G. & Jeffrey, R. (1974). *Computability and logic.* Cambridge university.

Charles, H. Bernett. "Turing Machine", in
http://www.seas.upenn.edu/~cit596/notes/turing.pdf

Church, Alonzo. (1956). *Introduction to Mathematical Logic.* Princeton, New Jersey: Princeton University Press.

Curtis, Brown. (2006). "Symbolic Logic Decidability", in:
http://www.trinity.edu/cbrown/logic/decidability.html.

-------------. (2004). "Church-Turing thesis", in *Wikipedia Encyclopedia*:
http://en.wikipedia.org/wiki/Church-Turing_thesis.

-------------. (2004). "Mathematical_function", in *Wikipedia Encyclopedia*:
http://en.wikipedia.org/w/index.php?title=Mathematical_function&redirect=no.

Cutland, N. J. (1989). *Computability: An Introduction to Recursive Function Theory*. Cambridge University.

Davis, Martin; Ron, Sigal; Elaine, J. Weyuker. (1994). *Computability, Complexity, and Languages: Fundamentals of Theoretical Computer Science.* Elsevier.

Danziger, P. "Russells Paradox and the Halting Problem": Ryerson University. MTH 314 - 5.4;
http://www.scs.ryerson.ca.

Enderton, Herbert B. "Turing and Computability", in
http://www.math.ucla.edu/~hbe/turing.pdf.

Epstein, Richard L. & Walter, A. Carnielli. (1989). *Computability Computable Function, Logic and Foundation of Mathematics.* California: Wadsworth & Brooks.

Fairtlough, Matt. (2003). "Introduction to Recursive Function Theory", in:
http://www.dcs.shef.ac.uk/~matt/teaching/04/com2030/lectures/tomlect13.pdf.

Fitting, Melvin. (1987). *Computability Theory, Semantics and Logic Programming.* Oxford University Press.

-------------. (2004). "Halting problem", in *Absolute Astronom Encyclopedia:*
http://www.absoluteastronomy.com/encyclopedia/h/ha/halting_problem.htm.

Hilbert, David. (1900-1904). *Grundlangend der geometri.* ed. by Griffor Edward R. "Hand book of computability Theory". Elsevier.

Jones, Neil D. (1997). "Computability and Complexity". *Massachusetts Institute of Technology*:
http://www.diku.dk/undervisning/2005f/729/Chapter1Printout.pdf.

Labor, low. Halting Problem, in "Unsolvability of the Halting Problem":
http:/google.com/Scolarship/Unsolvablity of the Halting problem.

Lewis. (2005). *Solvability and the Halting Problem.* University of Kentucky: College of Engineering.

Lyndon, Roger C, & Schupp, Paul E. (2001). *Combinatorial Group Theory.* Springer.

Melvin, Fitting. (1987). *Computability Theory, Semantics and Logic Programming.* Oxford University Press.

Mendelson, E. (1997). *Introduction to Mathematical Logic.* fourth edition, Chapman & Hall.

Minsky, Marvin. (2004). "Register Machine", in *Absolute Astronomy Encyclopedia*:
http://www.absoluteastronomy.com/encyclopedia/m/ma/marvin_minsky.htm.

Minsky, Marvin. *Facts, Info, and Encyclopedia Article*:
http://www.absoluteastronomy.com/encyclopedia/r/re/register_machine.htm.

Nolt, John. (1997). *Logics.* University of Tennessee, Knoxville: Wadsworth publishing company.

-------------. (1999-2005). "Primitive Recursive Function", in *Wolfram Research*:
http://mathworld.wolfram.com/PrimitiveRecursiveFunction.html.

-------------. "Recursive Functions", in *Stanford Encyclopedia of Philosophy*:
http://plato.stanford.edu/entries/recursive-functions.

-------------. (2004). "Register machine", in *Absolute Astronomy Encyclopedia*:
http://www.absoluteastronomy.com/encyclopedia/r/re/register_machine.htm.

Richard L. Epstein. & Walter, A. Carnielli. (1970). *Computability*, second edition.

Salomaa, Arto. (2003). *Computation and Automata.* Cambridge University Press.

Siqueira, Marcelo. (2004). "Theory of Computation", in: www.research.ibm.com/people/b/bennetc/TM2.pdf.

Sommerhald, R. Westrhenen S.C.Van. (1988). *The Theory of Computability.* University of Delft–Adison–Wesley Publishing Company.

Stanford Encyclopedia of Philosophy. (2005). "recursive-functions", in: http://plato.stanford.edu/entries/recursive-functions

Stuart J. Russell. & Peter Norvig. (1995). *Artificial Intelligence: A Modern Approach.* Prentice-Hall, Inc A Simon & Schuster Company Englewood Cliffs: New Jersey.