استفاده از ارضاپذیری در برنامه‌ریزی زمانی

سید کمال‌الدین غیاثی
ghiathi@mehr.sharif.edu
دانشکده‌ی مهندسی کامپیوتر
دانشگاه صنعتی شریف

غلامرضا قاسم‌ثانی
sani@sharif.edu
دانشکده‌ی مهندسی کامپیوتر
دانشگاه صنعتی شریف

چکیده

برنامه‌ریزی یکی از مباحث هوش مصنوعی است که در آن دنباله‌ای از کنش‌ها برای رسیدن به یک هدف از پیش‌تعیین شده بصورت خودکار تولید می‌شود. اکثر برنامه‌ریزهای سنتی برای ساده‌سازی مساله‌ی برنامه‌ریزی فرض می‌کنند که کنش‌ها بصورت آنی اجرا می‌شوند. یکی از معدود روش‌های برنامه‌ریزی که در آن با مساله‌ی زمان به‌صورت جامع برخورد شده‌است، برنامه‌ریزی زمانی آلن است که به دلیل پیچیدگی بسیار زیاد در عمل غیر قابل استفاده است. از طرف دیگر روش‌های نوین برنامه‌ریزی مانند SatPlan علیرغم کارایی بسیار بالا هنوز مفهوم زمان را در نظر نمی‌گیرند. در این مقاله روش جدیدی معرفی می‌گردد که برای افزایش کارایی برنامه‌ریزی زمانی آلن، ابتدا مساله را به یک مساله‌ی ارضای محدودیت تبدیل می‌کند و سپس آنرا از طریق روش‌های تست ارضاپذیری حل می‌نماید. در این مقاله نشان می‌دهیم که اگر روش آلن را اندکی تغییر دهیم، پیچیدگی حل مساله به‌طور قابل ملاحظه‌ای کاهش می‌یابد.

کلمات کلیدی: برنامه‌ریزی زمانی، ارضاپذیری.

۱) مقدمه

قبل از آنکه کاتز و سلمن [1] مقاله‌ی خود را در رابطه با برنامه‌ریزی[1] ارضاپذیری[2] ارائه کنند، به مساله‌ی برنامه‌ریزی به صورت یک مساله‌ی استنتاجی[3] نگاه می‌شد. در نتیجه اغلب برنامه‌ریزهای سنتی معمولا در منطق مرتبه‌ی اول به حل مساله می‌پردازند (به عبارت دیگر از یکسان‌سازی[4] استفاده می‌کنند). در دهه‌ی گذشته دو برنامه‌ریز جدید با نام‌های SatPlan[1,2] و GraphPlan[3]، به برنامه‌ریزی در منطق گزاره‌ای بازگشتند که باعث افزایش شدید سرعت برنامه‌ریزی شد. برنامه‌ریز SatPlan که اولین برنامه‌ریز از این نوع است در بخش دوم مقاله بررسی می‌شود.

یکی از جنبه‌هایی که انسان‌ها در برنامه‌ریزی در دنیای واقعی در نظر می‌گیرند، مساله‌ی زمان است. اغلب روش‌های برنامه‌ریزی

---

[1] Planning
[2] Satisfiability planning
[3] Deductive
[4] Unification

در مورد زمان پیش‌فرض‌های ساده‌کننده‌ای را در نظر می‌گیرند. مثلا فرض می‌شود که کنش‌ها به‌صورت آنی[1] اجرا می‌شوند. با این فرض روابط زمانی بین هر دو کنش به روابط ساده‌ی «قبل» ، «بعد» و یا «همزمان» تبدیل می‌گردد. اما چون در جهان واقعی کنش‌ها آنی نیستند، می‌توانند با یکدیگر رابطه‌ای غیر از این روابط ساده داشته باشند (رابطه‌ی بین ریزدستورالعمل‌ها را در یک ریزپردازنده در نظر بگیرید). در برخی از برنامه‌ریزهای قدیمی‌تر، تلاش شده‌بود تا مفهوم زمان با برنامه‌ریزی ترکیب شود[4,5]. هرچند زمان می‌تواند قدرت بیان[2] برنامه‌ریزها را برای توصیف مسایل افزایش دهد، ولی اکثر برنامه‌ریزهای زمانی موجود نتوانسته‌اند (و یا در یک مصالحه با سرعت، نخواسته‌اند) از تمام این قدرت بیان بهره برند. برخی از برنامه‌ریزها، زمان را تنها به صورت عددی می‌پذیرند[5] و برخی دیگر اجازه می‌دهند که زمان به صورت رابطه‌ای (مثلا روابط قبل از، پس از و همزمان با) بین چند گزاره و کنش[3] بیان شود[4]. از طرف دیگر برخی برنامه‌ریزها تنها با زمان‌های گسسته کار می‌کنند[1] و برخی دیگر اجازه‌ی کار با اعداد حقیقی را نیز می‌دهند[6]. همچنین در برخی سیستم‌ها طول مجاز کنش‌ها ثابت (ولی برای هر کنش متفاوت) است و در برخی دیگر در یک بازه[4] تغییر می‌کند. تاکنون روش آلن[4] از این حیث دارای بیشترین قدرت بیان است. آلن با ارائه‌ی مدلی برای برنامه‌ریزی که مبتنی بر یک منطق زمانی است، روشی برای وارد کردن زمان در برنامه‌ریزی ارائه نمود. اما مشکل اصلی روش آلن پیچیدگی بسیار بالای آن است که باعث می‌شود برای مسائل واقعی غیر قابل استفاده گردد. این مدل در بخش سوم مقاله بررسی می‌شود.

شباهت بسیاری بین دو برنامه‌ریز TIMELOGIC[5] و SatPlan[6] وجود دارد. برای مثال هر دو به اطلاعاتی بیش از اطلاعات داده‌شده در نمایش STRIPS[7] نیاز دارند. ما در این مقاله یک روش برنامه‌ریزی زمانی ارائه می‌کنیم که مساله‌ی برنامه‌ریزی را با روش ارضای محدودیت حل می‌کند. این روش در برنامه‌ریزی که آنرا TSat[7] نامیده‌ایم، پیاده‌سازی شده است. TSat یک ترکیب از کنش‌ها را انتخاب می‌کند و پس از یک سری تست مقدماتی، اگر مشخص شود که ترکیب انتخابی از کنش‌ها شانس آن را دارد که حل مساله باشد، آن را به یک فرمول CNF[8] تبدیل می‌کند. اگر این ترکیب پاسخ مساله باشد، این فرمول CNF ارضا پذیر است و می‌توان از مقدار متغیرها در حالت ارضا شده، پاسخ مساله‌ی برنامه‌ریزی را به دست آورد. الگوریتم برنامه‌ریزی Tsat در بخش چهارم مقاله تشریح می‌گردد.

در تبدیل مساله‌ی برنامه‌ریزی به فرمول CNF روش‌های مختلفی وجود دارد. این روش‌ها در کارایی با یکدیگر متفاوتند. برای مثال می‌توان مساله را براساس جبر بازه‌ها[9][4] یا بر اساس جبر نقطه‌ها[10][8] به فرمول CNF تبدیل کرد. این دو روش از نظر کارایی با یکدیگر متفاوتند. در بخش پنجم روش‌های مختلف کد کردن و میزان کارایی آنها بررسی می‌شود و در بخش ششم مثالی تجربی از کارایی این روش‌ها ارائه می‌شود.

۲) مروری بر برنامه‌ریز SatPlan

مسائل برنامه‌ریزی را می‌توان به عنوان مسائل ارضای محدودیت[11] در نظر گرفت[9]. همچنین ارضاپذیری[12] به‌عنوان نوع

---

[1] Instantaneous
[2] Expressiveness
[3] Action
[4] Interval

[5] TIMELOGIC نام برنامه‌ریز آلن است

[6] SatPlan اولین برنامه‌ریز مبتنی بر برنامه‌ریزی ارضاپذیری است.

[7] Temporal Satisfiability
[8] Conjunctive Normal Form.
[9] Interval Algebra
[10] Point Algebra
[11] Constraint satisfaction
[12] Satisfiability

خاصی از نمایش مسایل ارضای محدودیت توجه محققین بسیاری را به خود جلب کرده است. در سال ۱۹۹۲ کاتز برای آنکه برتری یک الگوریتم تست ارضاپذیری به نام GSAT[1] را نشان دهد، آن را به‌روی مساله‌ی برنامه‌ریزی (در برنامه‌ریزی به نام SatPlan) اعمال کرد و باعث شد دامنه‌ی مسائلی که برنامه‌ریزها قادر به حل آن بودند، گسترش یابد. این موفقیت تاکنون نیز ادامه داشته است به گونه‌ای که هنوز سریع‌ترین برنامه‌ریزها حداقل در بخشی از فرآیند حل مساله‌ی خود، از تست ارضا پذیری استفاده می‌کنند[10]. شاید یکی از دلایل موفقیت این روش، بهره‌گیری از کار زیادی است که در زمینه‌ی حل مساله‌ی SAT انجام شده است. در این بخش SatPlan را که اولین برنامه‌ریز مبتنی بر ارضاپذیری است، بررسی می‌کنیم.

SatPlan یک مساله‌ی برنامه‌ریزی را به عنوان ورودی می‌گیرد و (با این فرض که طول پاسخ برنامه از n بیشترنیست) یک فرمول CNF[1] معادل آن می‌سازد. منظور از معادل این است که اگر فرمول CNF ارضا پذیر باشد، جواب مساله‌ی برنامه‌ریزی را می‌توان در زمان چندجمله‌ای[2] از مقدار متغیرها در حالتی که CNF ارضا شده است، استخراج کرد؛ اما اگر مساله‌ی برنامه‌ریزی پاسخی نداشته باشد، فرمول CNF نیز ارضا پذیر نیست. SatPlan ابتدا فرض می‌کند که پاسخی به طول صفر وجود دارد. اگر فرمول CNF به‌دست‌آمده ارضا پذیر نبود، فرض می‌کند که پاسخی به طول یک وجود دارد و سعی می‌کند پاسخ را بیابد. برای افزایش سرعت، SatPlan تنها طول‌هایی را در نظر می‌گیرد که توانی از ۲ هستند (البته ممکن است پاسخی که به دست می‌آید از نظر طول کمینه نباشد). این عمل تا یافتن پاسخ مساله‌ی برنامه‌ریزی و یا بیشتر شدن طول فرض‌شده برای پاسخ از یک حد آستانه‌ای مشخص ادامه می‌یابد.

روش‌های متعددی برای کد کردن یک مساله به صورت فرمول CNF وجود دارد[11]. در اینجا یک روش ساده را شرح می‌دهیم[1]. فرض کنیم که اندازه پاسخ مساله d باشد. آنگاه d کنش وجود دارند که دارای یک ترتیب کامل[3] هستند (این روش کدگذاری اجازه‌ی هم‌زمانی کنش‌ها را نمی‌دهد). در هر لحظه یک کنش انجام می‌شود و برخی از گزاره‌ها درست هستند. متغیرهای CNF، انجام کنش یا درستی گزاره‌ها در هر یک از d لحظه است. حال که متغیرهای CNF مشخص شدند، می‌توان محدودیت‌های زیر را به صورت یک فرمول CNF در آورد.

۱. اگر کنش p در لحظه‌ی t انجام شود، آنگاه پیش‌شرط‌های آن در لحظه‌ی t-1 و نتایج آن در لحظه‌ی t درست هستند.
2. در هر لحظه‌ی دقیقا یک کنش انجام می‌شود.
۳. شرایط اولیه در لحظه‌ی صفر درست هستند.
٤. اهداف در لحظه‌ی d+1 درست هستند.
٥. اگر گزاره‌ای در لحظه‌ی t-1 درست باشد و کنشی که در لحظه‌ی t-1 انجام شده است آن را حذف نمی‌کند، آنگاه گزاره‌ی مزبور در لحظه‌ی t نیز درست خواهد بود.
٦. محدودیت‌های دامنه[4]. مثلا در دامنه‌ی دنیای مکعب‌ها[5] که یک محیط کلاسیک برای برنامه‌ریزی محسوب می‌شود، گزاره‌های ON(a,b) و ON(a,c) نمی‌توانند در یک لحظه درست باشند. به این معنی که در یک لحظه یک مکعب نمی‌تواند بر روی دو مکعب مختلف قرار داشته باشد.

این محدودیت‌ها می‌توانند به صورت گزاره‌های منطقی نوشته شوند. برای مثال برای بیان اینکه در هر لحظه حداقل یک کنش

---

[1] یک فرمول CNF یک گزاره منطقی است که به صورت یک ترکیب عطفی از ترکیب فصلی حروف ساخته می‌شود. یک حرف می‌تواند یک متغیر و یا نقیض آن باشد. برای مثال $(a \vee b \vee c \vee d) \wedge (\neg a \vee d) \wedge (b \vee \neg d)$ یک فرمول CNF است.

[2] Polynomial
[3] Total order
[4] Domain constraints
[5] Blocks world

انجام می‌شود، می‌توانیم بنویسیم:

$$(a_1(1)\vee a_2(1)\vee...\vee a_n(1))\wedge(a_1(2)\vee a_2(2)\vee...\vee a_n(2))\wedge...\wedge(a_1(d)\vee a_2(d)\vee...\vee a_n(d))$$

که n تعداد کنش‌ها است. در فرمول فوق هر $a_i(j)$ یک متغیر است.

فرمول CNF پس از ساخته شدن به یک الگوریتم تست ارضای محدودیت (مانند GSAT[1] ، WalkSat [12] و Tableau[13]) داده می‌شود تا مشخص گردد که آیا فرمول CNF ارضاپذیر است یا نه و اگر هست، به ازای چه مقادیری برای متغیرها فرمول ارضا می‌شود. به دست آوردن برنامه از روی مقدار متغیرها در حالت ارضا شده بسیار ساده است، زیرا فرمول در حالت ارضا شده نشان می‌دهد که در هر لحظه چه کنشی انجام می‌شود.

۳) برنامه‌ریز زمانی آلن (TIMELOGIC)

آلن مدلی را برای برنامه‌ریزی ارائه کرده است که منحصرا مبتنی بر مفهوم زمان است. در این مدل کنش‌ها و گزاره‌ها با بازه‌های زمانی[1] مربوط به خود منتسب می‌شوند؛ هر کنش (گزاره) در بازه‌ی زمانی مربوط به خود انجام می‌شود (درست است). جدول ۱ ارتباط‌های ممکن بین بازه‌ها را نشان می‌دهد.

جدول ۱: سیزده رابطه‌ی ممکن بین دو بازه

Relation	Symbol	Symbol for inverse	Pictorial example
X before Y	<	>	XXX YYY
X equal Y	=	=	XXX YYY
X meets Y	m	mi	XXXYYY
X overlaps Y	o	oi	XXX YYY
X during Y	d	di	XXX YYYYYYY
X starts Y	s	si	XXX YYYYY
X finishes Y	f	fi	XXX YYYYYY

در این روش در هنگام معرفی کنش‌های موجود به برنامه‌ریز، روابط بازه‌های مربوط به هر کنش نیز مشخص می‌گردد.. برای مثال با استفاده از روش نمایش STRIPS،کنش‌های MOVE و MOVTBL در دنیای مکعب‌ها چنین نمایش داده می‌شوند:

MOVE (x,y,z)

Preconditions: CLEAR(x), CLEAR(z),ON(x,y)

Effects: ON(x,z), CLEAR(y), ¬CLEAR(z) , ¬ON(x,y)

MOVTBL[2] (x,y)

Preconditions: CLEAR(x), ON(x,y)

Effects: ON(x,table) , CLEAR(y), ¬ON(x,y)

که در آن preconditions شرایطی است که پیش از انجام کنش باید برقرار باشند و effects نتایجی را که پس از اجرای

[1] Temporal intervals
[2] Move to table

کنش پدیدار می‌شوند، بیان می‌کند. کنش MOVE(x,y,z) مکعب x را از روی y بر روی مکعب z قرار می‌دهد و کنش MOVTBL(x,y) مکعب x را از روی مکعب y بر روی میز قرار می‌دهد. گزاره ON(x,y) به معنای قرار داشتن x بر روی y است و گزاره CLEAR(x) به این معنا است که هیچ مکعبی بر روی x قرار ندارد. در مدل زمانی آلن، این کنش‌ها به صورت زیر تعریف می‌شوند:

If MOVE (x,y,z) occurs over time Sxyz then
CLEAR(x) holds over time Cx1, Such that Cx1 *meets* Sxyz, and
CLEAR(z) holds over time Cz, Such that Cz *finishes* Sxyz, and
ON(x,y) holds over time Oxy, Such that Oxy *(o s d)* Sxyz, and
CLEAR(y) holds over time Cy, Such that Sxyz *(o fi di)* Cy, and
CLEAR(x) holds over time Cx2, Such that Sxyz *meets* Cx2, and
ON(x,z) holds over time Oxz, Such that Sxyz *overlaps* Oxz.

If MOVTBL (x,y) occurs over time Sxyt then
CLEAR(x) holds over time Cx1, Such that Cx1 *meets* Sxyt, and
ON(x,y) holds over time Oxy, Such that Oxy *(o s d)* Sxyt, and
CLEAR(y) holds over time Cy, Such that Sxyt *(o fi di)* Cy, and
CLEAR(x) holds over time Cx2, Such that Sxyt *meets* Cx2, and
ON(x,table) holds over time Oxt, Such that Sxyt *overlaps* Oxt.

قبل از بیان الگوریتم برنامه‌ریزی آلن، چند واژه را تعریف می‌کنیم. منظور از دوختن[1] دو بازه، این است که رابطه‌ی بین آنها را به مساوی محدود کنیم. اگر برخی از اهداف یا پیش‌شرط‌های کنش‌ها هنوز به بازه یک گزاره حالت اولیه یا نتیجه یک کنش دوخته نشده‌اند، می‌گوییم که یک فاصله‌ی سببی[2] وجود دارد و آن هدف یا پیش‌شرط کنش هیچ توجیهی ندارد (توجیه نشده است).

برنامه‌ریزی به این صورت انجام می‌شود که در ابتدا صورت مساله (یعنی گزاره‌های حالت شروع و حالت هدف) به صورت تعدادی بازه در نظر گرفته می‌شوند. دو بازه‌ی ویژه، I (حالت شروع) و G (حالت هدف) نیز وجود دارند. بازه‌های گزاره‌های حالت شروع شامل[3] I و بازه‌های گزاره‌های حالت هدف شامل G هستند. برنامه‌ریز دائما فرآیند یافتن فاصله‌های سببی و حذف آنها را با وارد کردن کنش‌های جدید و یا دوختن دو بازه‌ی مناسب ادامه می‌دهد. پس از دوخته شدن دو بازه، روابط بین آنها به رابطه‌ی مساوی محدود می‌شود. این محدودیت می‌تواند انتشار یابد و روابط دیگر را نیز محدودتر کند. اگر مجموعه‌ی روابط ممکن بین دوبازه تهی شود، برنامه‌ریز عقب‌گرد[4] می‌کند. از طرف دیگر اگر هیچ فاصله‌ی سببی‌ای وجود نداشته باشد، مساله حل شده است.

در انتشار محدودیت‌ها علاوه بر آنچه که به جبر بازه‌ها موسوم است، از دو دسته محدودیت دیگر نیز استفاده می‌شود: محدودیت‌های دامنه[5] و محدودیت‌های گزاره‌ای[6].

محدودیت‌های دامنه:

در هر دامنه محدودیت‌هایی وجود دارد که در استنتاج استفاده نمی‌شوند. این محدودیت‌ها نتیجه‌ی حالت اولیه و

---

[1] آلن از واژه Collapse که به معنی فروریختن است به عنوان برقراری رابطه مساوی بین دو بازه استفاده کرده است. لیکن در اینجا با توجه به مفهوم عمل انجام شده، واژه دوختن ترجیح داده شده است.

[2] Causal gap

[3] می‌گوییم بازه A شامل بازه B است هرگاه رابطه‌ی A(di si fi) B برقرار باشد.

[4] Backtrack

[5] Domain Constraints

[6] Propositional Constraints

تعریف کنش‌ها هستند[3]. برای مثال در دنیای مکعب‌ها اگر با حالتی شروع کنیم که هیچ مکعبی ،مانند a، نباشد که گزاره‌های CLEAR(a) و ON(b,a) در حالت اولیه درست باشند، آنگاه به حالتی نخواهیم رسید که در آن گزاره‌های CLEAR(a) و ON(b,a) همزمان درست باشند. به این اطلاعات اضافی در هنگام استنتاج نیازی نیست، ولی استفاده از آنها در هنگام استفاده از روش ارضاپذیری (یا ارضای محدودیت) ضروری است.

این نکته‌ی جالبی است که با وجود اینکه TIMELOGIC مساله‌ی برنامه‌ریزی را با استنتاج حل می‌کند، ولی به این علت که اعمال انتشار محدودیت و ارضای محدودیت را انجام می‌دهد، مجبور به استفاده از محدودیت‌های دامنه است.

محدودیت‌های گزاره‌ای:

دو بازه که به گزاره‌ی مشابهی منتسب شده باشند، تنها می‌توانند یکی از روابط قبل، بعد یا مساوی را داشته باشند. مثلا اگر بدانیم که در بازه I1 و I2 گزاره‌ی P درست است، آنگاه رابطه‌ی دو بازه‌ی I1 و I2 بصورت I1 (< = >) I2 است.

٤) برنامه‌ریزی زمانی به عنوان ارضای محدودیت

در این بخش روش جدیدی که از ادغام روش‌های برنامه‌ریزی SatPlan و TIMELOGIC بوجود آمده است، تشریح می‌گردد. با توجه به مطالب مطرح شده در بحث‌های قبل، شباهت‌های زیر بین TIMELOGIC و SatPlan دیده می‌شود:

۱. هر دو کنش‌ها را به صورت یک گزاره شرطی تعریف می‌کنند که در آن اجرای کنش منجر به درستی پیش‌شرط‌های کنش قبل از اجرا و نیز درستی نتایج کنش پس از اجرای آن می‌شود.
۲. هر دو فرقی بین پیش‌شرط و نتیجه قائل نیستند. البته در TIMELOGIC در ابتدای ورود یک کنش به برنامه، پیش‌شرط‌ها فاقد توجیه و نتایج توجیه‌شده فرض می‌شوند.
3. هر دو به اطلاعات بیشتری از آنچه در نمایش STRIPS داده می‌شود، نیاز دارند. برای مثال SatPlan به یک حالت شروع کامل (یعنی وضعیت تمام گزاره‌ها باید مشخص باشد) و هر دو به محدودیت‌های دامنه نیاز دارند.

تفاوت‌های زیر نیز بین این دو برنامه‌ریز قابل تشخیص است:

۱. TIMELOGIC مساله را در منطق مرتبه‌ی اول حل می‌کند، در حالی که SatPlan مساله را در منطق گزاره‌ای حل می‌کند.
۲. در TIMELOGIC مساله با استنتاج حل می‌شود، در حالی که SatPlan مساله را با استفاده از ارضاپذیری حل می‌کند.

در این بخش ما یک برنامه‌ریز جدید به نام TSat معرفی می‌کنیم که از نقطه نظر روابط زمانی ممکن بین کنش‌ها مشابه TIMELOGIC است، ولی مساله‌ی برنامه‌ریزی را با کمک ارضای محدودیت حل می‌کند. هدف از این ادغام، افزایش سرعت برنامه‌ریزی آلن ضمن حفظ قدرت بیان آن است. برای یافتن پاسخ، یک جستجوی عرض اول[1] انجام می‌دهیم. یعنی ابتدا پاسخی با طول صفر (طول پاسخ برابر با تعداد کنش‌های آن است) جستجو می‌شود و در صورت عدم موفقیت، به طول پاسخ یکی اضافه می‌شود. این کار تا زمانی که پاسخ پیدا شود و یا طول پاسخ از یک مقدار آستانه‌ای از پیش تعیین شده بیشتر شود، ادامه می‌یابد. برای هر طول مشخص پاسخ، تمام ترکیبات (نه ترتیب) ممکن کنش‌ها تولید می‌شوند. سپس به‌ازای هر ترکیب ممکن از کنش‌ها یک فرمول CNF می‌سازیم. فرمول CNF را چنان می‌سازیم که تنها اگر مساله برنامه‌ریزی را بتوان با

[1] Breadth first

استفاده از کنش‌های انتخاب شده حل کرد، این فرمول ارضاپذیر باشد. همچنین می‌توان با توجه به مقادیر متغیرهای فرمول CNF در حالت ارضا شده، پاسخ مساله برنامه‌ریزی را به‌دست آورد (یعنی ترتیب نیز مشخص می‌شود).

هر فرمول CNF توسط محدودیت‌های زیر ساخته می‌شود:

1. بازه‌ی I قبل از بازه‌های دیگر است و یا آنها را ملاقات[1] می‌کند. همچنین بازه‌ی G پس از دیگر بازه‌ها است و یا توسط آنها ملاقات می‌شود.
2. بازه‌ی I بازه‌های مربوط به حالت اولیه را ملاقات می‌کند و گزاره‌های حالت هدف بازه‌ی G را ملاقات می‌کنند.
3. محدودیت‌های بین بازه‌های کنش‌ها که در تعریف کنش‌ها داده شده‌اند.
4. محدودیت‌های دامنه.
5. محدودیت‌های گزاره‌ای.
6. بازه‌های گزاره‌های حالت هدف و بازه‌های نسبت داده شده به پیش‌شرط‌های کنش‌ها باید توجیه شوند.
7. از بین ۱۳ رابطه‌ی ممکن بین هر دو بازه حداقل یکی درست است.
8. از بین ۱۳ رابطه‌ی ممکن بین هر دو بازه حداکثر یکی درست است.
9. محدودیت‌های انتشار (جبر بازه‌ها). برای مثال اگر a<b و b<c درست باشد،آنگاه a<c نیز برقرار است.

در خصوص بند ۸ ذکر این نکته ضروری است که در مدل آلن یک برنامه مشخص (با ترتیب کامل) به دست نمی‌آید، بلکه روابط ممکن بین گزاره‌ها مشخص می‌شود و ادعا شده است که با عمل انتشار محدودیت، اگر هیچ رابطه‌ای تهی نشود، آنگاه حداقل یک برنامه مشخص وجود دارد. ولی TSat یک برنامه مشخص را تولید می‌کند. حتی اگر ادعای فوق درست باشد، مدل آلن مزیتی بر TSat ندارد زیرا پاسخ آلن به اندازه کافی کلی نیست. یعنی ممکن است در یک مثال عددی که طول بازه‌ها مشخص است مساله جواب داشته باشد، ولی پاسخ سیستم آلن آن مثال را نپذیرد. علت این است که مدل آلن تنها برای یک روش انتخاب کنش‌ها، فرم کلی تولید می‌کند.

پس از ساختن فرمول CNF آن را به نرم‌افزار tableau[2] می‌دهیم تا ارضاپذیری آن چک شود. قبل از ساختن کامل فرمول CNF، برخی تست‌های اولیه انجام می‌شود. مهم‌ترین این تست‌ها اطمینان از این مطلب است که برای هر فاصله‌ی سببی، حداقل یک بازه با همان گزاره وجود دارد. این تست بسیار سریع است و مانع تولید فرمول CNF جز در تعداد کمی از ترکیبات کنش‌ها می‌شود.

۵) مقایسه‌ای بین روش‌های مختلف کد کردن

روش‌های مختلفی برای تبدیل یک مساله به فرمول CNF معادل وجود دارد. به هر یک از این روش‌ها یک روش کد کردن می‌گویند. با محاسبه تعداد عباراتی که توسط هر یک از محدودیت‌های معرفی‌شده در بخش ٤ ایجاد می‌شود، می‌توان نشان داد که مهمترین عامل در اندازه‌ی فرمول CNF و سختی مساله، مورد نهم این محدودیت‌ها است[3]. تعداد محدودیت‌های انتشار برابر I^3R^2 است که در آن I تعداد بازه‌ها و R تعداد روابط ممکن بین دو بازه است. در پیاده‌سازی TSat[4] که مبتنی بر جبر

---

[1] Meet

[2] روش ما برای حل مساله ناسازگاری سازمن بیش از یک میلیون عبارت تولید می‌کند که کد tableau قادر به مدیریت آن نیست. به همین دلیل کد مزبور اندکی تغییر کرد.

[3] اگر تمام عبارات دارای حداکثر دو متغیر باشند مساله در زمان چند جمله‌ای حل می‌شود[16]. بنابراین می‌توان گفت که عبارات با بیش از ۲ متغیر مساله را سخت می‌کنند.

[4] TSat هم براساس جبر بازه‌ها و هم بر اساس کدکردن طبیعی بهبودیافته پیاده‌سازی شده است.

بازه‌ها است، R برابر ۱۳ می‌باشد. از آنجا که هر محدودیت یک گزاره‌ی شرطی می‌باشد که بخش نتیجه‌ی آن به صورت یک عبارت فصلی است، هر محدودیت انتشار به یک عبارت در فرمول CNF تبدیل می‌شود زیرا:

$$a \wedge b \rightarrow c_1 \vee \ldots \vee c_k \Leftrightarrow \neg a \vee \neg b \vee c_1 \vee \ldots \vee c_k$$

چون محدودیت‌های انتشار بیشترین تاثیر را در سختی و بزرگی مساله دارند، روش‌های کد کردن را بر اساس تعداد محدودیت‌های انتشار و تعداد متغیر در هر عبارت بررسی می‌کنیم.

۵-۱) کد کردن به روش TIMELOGIC

در این کدگذاری R برابر ۱۳ است. با تغییر نام I به n به $169n^3$ محدودیت انتشار می‌رسیم. می‌توان محدودیت‌هایی را که روابط بین ۲ بازه را به همان ۱۳ رابطه‌ی کلی محدود می‌کنند، حذف کرد. با انجام این کار تعداد محدودیت‌های انتشار به $166n^3$ می‌رسد. در این روش تعداد متغیرهای هر عبارت می‌تواند حتی ۱۲+۲ (۲ تا برای بخش پیش‌شرط و ۱۲ تا برای بخش نتیجه) باشد.

۵-۲) کد کردن طبیعی

انسان‌ها معمولا با سه رابطه‌ی < = > کار می‌کنند. اگر بازه‌ها را به لبه‌هایشان بشکنیم، به 2n لبه می‌رسیم. از طرفی تعدادی محدودیت برای اطمینان از اینکه لبه‌ی سمت چپ قبل از لبه‌ی سمت راست است، باید اضافه شود (این محدودیت‌های اضافه بسیار مفیدند، زیرا عباراتی با تنها یک متغیر را تشکیل می‌دهند که کمک بزرگی در ساده‌شدن فرمول CNF است). با این روش تعداد عبارات در فرمول CNF به $3^2(2n)^3$ یا $76n^3$ کاهش می‌یابد. اگر عباراتی که محدود کننده نیستند (بخش نتیجه‌ی آنها این است که دو نقطه می‌توانند هر یک از ۳ رابطه‌ی فوق را داشته باشند) حذف کنیم، تعداد عبارات به $7(2n)^3$ یا $56n^3$ کاهش می‌یابد. مزیت دیگر این روش، کاهش تعداد متغیرها در هر عبارت است. در TIMELOGIC عباراتی با ۱۴ متغیر داشتیم، در حالی که در روش کد کردن طبیعی تعداد متغیرهای هر عبارت حداکثر ۳ (۲ تا برای بخش پیش‌شرط و یکی برای تنها حالت ممکن بخش نتیجه) است. این نکته بسیار مهم است زیرا هرچند هنوز هیچ راه حلی بهتر از راه حل بدیهی با زمان $O(2^n)$ (n تعداد متغیرهای فرمول CNF است) برای حل مساله‌ی SAT پیدا نشده [14] ولی برای 3-SAT راه حل بهتری با زمان $O(1.476^n)$ ارائه شده است [15]. همچنین مدارکی وجود دارد که اگر P≠NP آنگاه k-SAT (مساله‌ی SAT برای حالتی که حداکثر تعداد متغیرها در هر عبارت k است) برای k های کوچکتر سریع‌تر حل می‌شود[14].

۵-۳) کد کردن طبیعی بهبود یافته

می‌توان روابط < = > را با رابطه‌ی ≤ نیز نشان داد. برای مثال اگر هر دو رابطه‌ی a≤b و b≤a درست باشند، آنگاه رابطه‌ی a=b برقرار است. به این ترتیب تعداد عبارات انتشار به $(2n)^3$ یا $8n^3$ کاهش می‌یابد. مزیت دیگر این روش، افزایش تعداد عبارات تک متغیری است؛ زیرا اکنون رابطه‌ی = به صورت ترکیب عطفی روابط ≤ نشان داده می‌شود. همانطور که دیده می‌شود، اندازه‌ی مساله در این روش کد کردن۲۰ بار کمتر از روش کد کردن بر مبنای TIMELOGIC است. با توجه به اینکه اندازه‌ی مساله‌ی CNF رابطه‌ی نمایی با زمان دارد، می‌توان گفت که کاهش طول مساله بطور نمایی سرعت را افزایش می‌دهد.

۶) یک مثال تجربی

در این بخش به عنوان نمونه نشان می‌دهیم که چگونه TSat مساله‌ی موسوم به ناسازگاری سازمن[1] را حل می‌کند. صورت مساله‌ی ناسازگاری سازمن در شکل ۱ نشان داده شده است. نسخه‌ای از TSat که مبتنی بر کد کردن به روش TIMELOGIC است، این مساله را با استفاده از یک کامپیوتر 950MHz و 250MB حافظه، در ۳۵ ثانیه حل می‌کند. قبل از یافتن پاسخ، ۳ فرمول CNF تولید می‌شود که هر یک از حدود ۸ میلیون عبارت تشکیل می‌شوند. نسخه‌ای از TSat که مبتنی بر کد کردن طبیعی بهبودیافته است، این مساله را در ۳ ثانیه حل می‌کند. این‌بار تنها یک فرمول CNF که شامل چهارصد هزار عبارت است تولید می‌شود. شکل ۲ پاسخ مساله‌ی مزبور را که توسط TSat تولید شده است، نشان می‌دهد. کد این برنامه از طریق آدرس http://ce.sharif.edu/~ghiathi/SATEMP قابل دسترسی است.

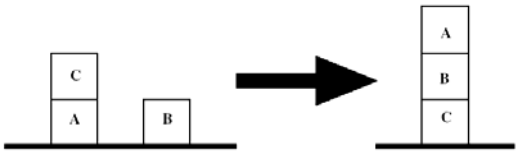
۷) نتیجه‌گیری

در این مقاله روشی ارائه شد که ترکیبی از ایده‌های دو برنامه‌ریز TIMELOGIC و SatPlan است. در این روش ضمن حفظ قدرت بیان TIMELOGIC (که SatPlan از آن بی‌بهره است) سرعت برنامه‌ریزی زمانی افزایش می‌یابد. هر چند مقایسه‌ی تجربی بین روش جدید و روش آلن به علت در دسترس نبودن کد TIMELOGIC انجام نشده است، اما در بخش ۵ بصورت تحلیلی نشان داده شد که TSat سریع‌تر از TIMELOGIC عمل می‌کند. در واقع عمده‌ی دلایل برتری TSat بر برنامه‌ریز آلن عبارتند از:

۱. TSat چون ابتدا محدودیت‌ها را به صورت یک فرمول CNF در می‌آورد، می‌تواند از ابتکاراتی مانند اصل شکست در ابتدا[2] در حل سریع مساله استفاده کند (در عمل سرعت برنامه‌ی tablau از همین ابتکارات نشات می‌گیرد).

۲. با تبدیل مساله‌ی برنامه‌ریزی به مساله‌ی ارضای محدودیت، می‌توان از روش‌هایی برای کد کردن استفاده کرد که اندازه‌ی کد CNF تولیدشده را بیش از ۱/۲۰ کاهش می‌دهد. توجه داریم که کاهش طول مساله‌ی SAT به صورت خطی به معنای افزایش سرعت به صورت نمایی است. همچنین دیدیم که حتی TIMELOGIC نیز در صورت استفاده از جبر نقطه‌ها به‌جای جبر بازه‌ها سریع‌تر خواهد بود.

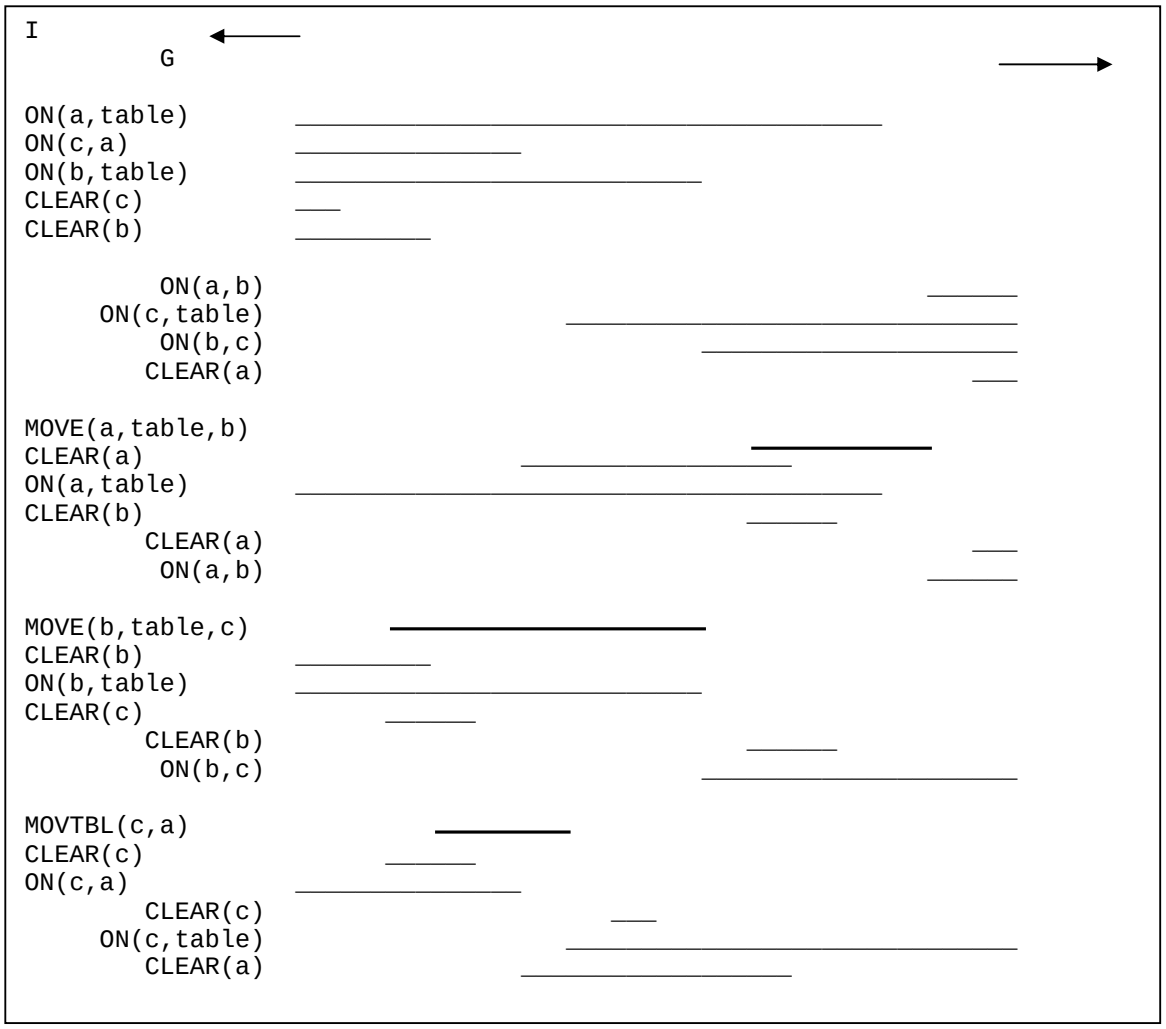
البته با وجود افزایش کارایی قابل توجه نسبت به TIMELOGIC، ولی TSat نیز در محیط واقعی قابل استفاده نیست. یکی از کارهای آتی این است که در ازای برخی‌ساده‌سازی‌ها، سرعت برنامه را به‌طور قابل ملاحظه‌ای افزایش دهیم. ایده اصلی این است که پیش‌شرط‌ها و نتایج کنش‌های جهان واقعی معمولا قبل و پس از کنش هستند و حداکثر با آن همپوشانی دارند. به عبارت دیگر معمول نیست که پیش‌شرطی پس از اجرای کنش بوجود آید و نتیجه‌ی کنش قبل از اجرای آن دیده شود. با این فرض می‌توان توجیه‌های دوری برای فواصل سببی را که در عمل یکی از عوامل مهم تلف شدن زمان بود، حذف کرد.



شکل ۱ : مساله‌ی ناسازگاری سازمن که با حالت شروع و هدف مشخص شده است.

---

[1] Sussman anomaly
[2] Fail first principle



شکل ۲: پاسخ TSat به مساله‌ی ناسازگاری سازمن

۸) مراجع

[1] Henry Kautz and Bart Selman. "Planning as Satisfiability". In proceedings of the 10th Europian Conference on Artificial Intelligence (ECAI 92), Vienna, Austria, August 1992.

[2] Henry Kautz and Bart Selman. "Pushing the Envelope: Planning, Propositional Logic, And Stochastic Search". *In proceedings of the 13th National Conference on Artificial Intelligence (AAAI- 96), Porland, OR*, 1996.

[3] Avrim L. Blum and Merrick L. Furst. "Fast Planning Through Graph Analysis". *Artificial Intelligence*, 90:281-300, 1997.

[4] James F. Allen and Johannes A. Koomen. "Planning using a temporal world model". *In Proceedings of the Eighth International Joint Conference on Artificial Intelligence.* 741-747. 1983.

[5] Steven A. Vere. "Planning in Time: Windows and Durations for Activities and Goals". *IEEE.* 1983.

[6] David E. Smith and Daniel S. Weld. "Temporal Planning With Mutual Exclusion

Reasoning". *IJCAI*. 1999.
[7] R. E. Fikes and N. J. Nilsson. "STRIPS: a new approach to the application of theorem proving to problem solving". *Artificial Intelligence*, 2(3-4):189-208. 1971.
[8] T. Dean and M. Boddy. "Incremental causal reasoning". *AAAI87*. 196-201. 1987.
[9] Matk Stefik. "Planning and Meta-Planning (MOLGEN: Part 2)". *Artificial Intelligence 16* 141-170. 1981.
[10] Henry Kautz and Bart Selman. "Unifying sat-based and graph-based planning". *In Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 318–325. Morgan Kaufmann, 1999.
[11] Michael D. Ernst, Todd D. Millstein and Daniel S. Weld. "Automatic SAT-Compilation of Planning Problems". *In proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI- 97), Nagoya, Aichi, Japan*, August 23-29, 1997.
[12] Bart Selman, Henry Kautz and B. Cohen. "Local search strategies for satisfiability testing". *Dimacs Series in Discrete Mathematics and Theoretical Computer Science.* 1996.
[13] James M. Crawford and Larry D. Auton. "Experimental results on the crossover point in satisfiability problems". *Proc. AAAI-93, Washington DC.*1993.
[14] Russel Impagliazzo and Ramamohan Paturi. "Complexity of k-SAT". IEEE Conference on Computational Complexity. 1999.
[15] Robert Rodosek. "A new approach to soling 3-satisfiability". *In Proceedings of the 3rd International Conference on Artificial Intelligence and Symbolic Mathematical Computation*, pages 197-212. Springer, LNCS 1138, 1996.
[16] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. Introduction to Algorithms (second edition). *MIT press*, page 1003. 2001.