DOR: 20.1001.1.2538161.1404.10.3.3.3

DOI: 10.22034/csj.2025.233054

پاییز ۱۴۰۴، دوره ۱۰، شماره ۳، پیاپی ۳۸

دریافت مقاله: ۱۴۰۴/۰۲/۱۳

پذیرش مقاله: ۱۴۰۴/۰۳/۱۸

نوع مقاله: پژوهشی

یادگیری تقویتی برای تطبیق خودکار عناصر در تبدیل مدل: کاهش پیچیدگی و افزایش مقیاس‌پذیری

دل‌آرام نیکبخت نصرآبادی

دانش‌آموخته گروه مهندسی کامپیوتر، دانشکده فنی و مهندسی، دانشگاه شهرکرد، شهرکرد، ایران

پست الکترونیکی: delaram.nikbakht@stu.sku.ac.ir

لیلا صمیمی دهکردی*

استادیار گروه مهندسی کامپیوتر، دانشکده فنی و مهندسی، دانشگاه شهرکرد، شهرکرد، ایران

پست الکترونیکی: samimi@sku.ac.ir

محمد احسان بصیری

دانشیار گروه مهندسی کامپیوتر، دانشکده فنی و مهندسی، دانشگاه شهرکرد، شهرکرد، ایران

پست الکترونیکی: basiri@sku.ac.ir

چکیده

این مقاله یک رویکـرد نوین مبتنی بر یادگیری تقویتی را برای تطبیق خودکار عناصر بین مدل‌های مبدا و مقصد در فرایند تبدیل مدل‌ها ارائه می‌دهد. این الگوریتم که با نام کیوموتور شناخته می‌شـود، با استفاده از جداول پاداش، نگاشـت‌های بهینه بین عناصر مدل‌هـا را یاد می‌گیرد. در روش پیشـنهادی، یـک جفت مدل نمونه شـامل مدل مبدا و مدل مقصـد به‌همـراه فرامدل‌های آن‌ها، بـه الگوریتم کیوموتور داده می‌شود. این الگوریتم سعی می‌کند در طول فرایند یادگیـری یک مدل میانی مبتنی بر فرامدل مقصد از روی عناصـر مدل مبدا ایجاد کند و این مـدل میانی را با مدل مقصد مربوط به نمونه ورودی مقایسه کرده و جدول پاداش مبتنی بر شباهت ساختاری را به‌روز می‌کند. الگوریتم یادگیری تقویتی کیوموتور نیاز به یادگیری زبان‌های تبدیل پیچیده (مانند ATL و ETL) را حذف کرده و تنها با یک جفت نمونه اولیه، قادر به تعمیم قوانین تبدیل به نمونه‌های مشابه اسـت. این ویژگی نه‌تنها پیچیـدگی فرایند تبدیل را کاهش می‌دهد و اسـتفاده برای کاربران غیرمتخصص را تسهیل می‌کند، بلکه امکان پردازش مدل‌های بسیار بزرگ (با بیش از ۶۰۰/۰۰۰ عنصر) را نیز فراهم می‌سـازد. نتایج ارزیابی نشان‌دهندۀ برتری این الگوریتم نسبت به روش‌های مرسوم در زمان اجرای کوتاه‌تر و مقیاس‌پذیری بالاتر اسـت. در ادامه، پیشنهادهایی برای توسعه آینده شامل افزایش دقت از طریق ادغام یادگیری عمیق، بهبود شناسـایی الگوهای پیچیده‌تر و طراحی سیستم ارزیابی خودکار کیفیت تبدیل ارائه شده است. این رویکرد می‌تواند به‌عنوان پایه‌ای برای پژوهش‌های آینـده در زمینه خودکارسـازی فرایندهای مدل‌سازی مورد استفاده قرار گیرد.

واژه‌های کلیـدی: یادگیری تقویتی، تبدیل مدل با مثال، مهندسی مبتنی بر مدل، خودکارسازی

* نویسندۀ مسئول

۱- مقدمه

در سال‌های اخیر، مهندسی نرم‌افزار مبتنی بر مدل به‌عنوان یک رویکرد مؤثر در توسعه سیستم‌های نرم‌افزاری مطرح شده است. این رویکرد بر مدل‌سازی به‌عنوان ابزاری کلیدی برای تحلیل، طراحی و توسعه نرم‌افزار تأکید دارد. در این رویکرد، مدل‌ها نه‌تنها برای مستندسازی سیستم، بلکه برای تولید خودکار کد، تحلیل سیستم و تبدیل مدل‌ها نیز مورد استفاده قرار می‌گیرند. مدل و فرامدل دو عنصر اصلی در این رویکرد می‌باشند. مدل‌ها که انتزاعی از سیستم واقعی هستند، فرایند توسعه نرم‌افزار را رو به جلو می‌رانند و فرامدل‌ها عناصر مدل‌سازی را توصیف می‌کنند و مدلی برای مدل‌ها هستند [۳]. تبدیل مدل یکی از مهم‌ترین جنبه‌های این رویکرد است که امکان تغییر مدل‌ها از یک نمایش به نمایش دیگر را فراهم کرده و موجب تسهیل ارتباط میان مراحل مختلف توسعه نرم‌افزار می‌شود. تبدیل مدل فرایندی است که مدل‌های مختلف را به هم تبدیل می‌کند و به‌عنوان قلب و روح این رویکرد شناخته می‌شود [٦].

در روش‌های سنتی، معمولا از یک زبان خاص تبدیل مدل برای نوشتن برنامه تبدیل استفاده می‌شد. استفاده از زبان تبدیل گرچه برای برخی موارد مؤثر است، اما چالش‌هایی دارد که عبارتند از:

۱. زمان‌بر بودن و مستعد خطا بودن: تعریف دستی قوانین تبدیل برای مدل‌های پیچیده یا در حال تکامل، نیازمند زمان و دقت بالایی است و احتمال بروز خطاهای انسانی در آن زیاد است.

۲. عدم انعطاف‌پذیری: قوانین تعریف شده معمولاً ثابت هستند و به‌راحتی قادر به سازگاری با تغییرات جدید در مدل‌ها نیستند.

برای غلبه بر این محدودیت‌ها، تبدیل مدل با مثال[1] به‌عنوان روشی جایگزین معرفی شده است. در این روش، قوانین تبدیل بر اساس الگوهای استخراج شده از جفت مدل‌های مبدا و مقصد به‌صورت نیمه‌خودکار تولید می‌شوند. گرچه این روش نسبت به روش‌های دستی کارآمدتر است، اما همچنان نیازمند تعریف نگاشت اولیه بین مدل‌های مبدا و مقصد است.

یکی از روش‌های نوین برای حل این چالش‌ها، استفاده از یادگیری تقویتی[2] در تبدیل مدل‌ها است. یادگیری تقویتی یک روش آزمون و خطا است که در آن یک عامل[3] با تعامل با محیط و دریافت پاداش یاد می‌گیرد که چگونه بهترین تصمیم‌ها را اتخاذ کند. در این پژوهش، تبدیل مدل با مثال به‌عنوان یک مسئله تصمیم‌گیری گام‌به‌گام، مدل‌سازی شده و از الگوریتم‌های یادگیری تقویتی برای استخراج خودکار قوانین تبدیل استفاده شده است.

رویکرد پیشنهادی در این مقاله، با نام «کیوموتور[4]» نیازی به زبان‌های تبدیل پیچیده ندارد و تنها با دریافت یک نمونه اولیه از تطبیق مدل‌ها، قادر است قوانین تبدیل را یاد گرفته و برای سایر مدل‌های مشابه (مبتنی بر فرامدل‌های نمونه اولیه) به‌صورت خودکار اعمال کند[5]. این ویژگی باعث می‌شود که فرایند تبدیل مدل‌ها سریع‌تر، دقیق‌تر و کم‌خطاتر باشد. همچنین، این روش قابلیت سازگاری با تغییرات مدل‌ها را دارد، زیرا عامل یادگیری تقویتی می‌تواند با دریافت بازخورد و اصلاح تدریجی عملکرد خود را بهینه کند. قوانین استخراج شده توسط این الگوریتم، در صورتی که مدل‌های جدید در چارچوب همان فرامدل اولیه باشند، به‌درستی قابل تعمیم هستند و دقت الگوریتم در این حالت حفظ می‌شود. همچنین، روش پیشنهادی در مقایسه با زبان‌های تبدیل معروف مانند زبان تبدیل اطلس (ATL) [۹] و زبان تبدیل اپسیلون (ETL) [۱۱] مقیاس‌پذیرتر بوده و زمان اجرای کمتری روی مدل‌های یکسان دارد. این موضوع در بخش ارزیابی نشان داده شده است.

در ادامه، مقاله به‌صورت زیر سازمان‌دهی شده است: بخش دوم کارهای مرتبط را بررسی می‌کند. در بخش

1-Model Transformation by Example

2-Reinforcement Learning

3-Agent

4-Q-learning MOdel TransfORmations (QMotor)

۵-کدها و مدل‌ها از پیوند زیر قابل بارگیری می‌باشند:
https://github.com/leilasamimi/QMotor

سوم، روش پیشنهادی ارائه می‌شود. در بخش چهارم، نتایج ارزیابی تجربی بررسی می‌گردد. در نهایت، در بخش پنجم، به بحث راجع به کارهای آتی پرداخته می‌شود.

۲- کارهای مرتبط

در این بخش، به بررسی پژوهش‌های پیشین مرتبط با تبدیل مدل، به‌ویژه روش‌های مبتنی بر هوش مصنوعی و یادگیری تقویتی پرداخته می‌شود. پژوهش‌های متعددی در زمینه بهینه‌سازی تبدیل مدل، خودکارسازی فرایند تبدیل و استفاده از یادگیری تقویتی انجام شده است.

شعرباف و همکاران [۱] روشی برای حل خودکار ناسازگاری‌ها در ادغام مدل‌ها ارائه کرده‌اند. آن‌ها با استفاده از الگوریتم‌های یادگیری تقویتی سعی کرده‌اند فرایند حل ناسازگاری‌ها را بهینه‌سازی کرده و میزان خودکارسازی را افزایش دهند. برای ارزیابی این رویکرد، آن‌ها یک سناریوی نمونه با نمودارهای کلاس UML طراحی کرده و دقت روش پیشنهادی خود را با الگوریتم‌های حریصانه و جستجو- محور مقایسه کرده‌اند. همچنین، از طریق یک آزمایش کاربری، میزان رضایت مدل‌سازان از تصمیمات اتخاذ شده در حل ناسازگاری‌ها را بررسی کرده‌اند.

باتوت و همکاران [۲] یک روش برنامه‌نویسی ژنتیک تک‌هدفه و چندهدفه را برای استخراج دانش خودکارسازی در مدل‌سازی نرم‌افزار پیشنهاد کردند. این روش به‌منظور بهبود فرایند مدل‌سازی از طریق تنوع اجتماعی در الگوریتم ژنتیک ارائه شده است. نتایج نشان داد که استفاده از این رویکرد، همگرایی سریع‌تر و بهبود تعمیم‌پذیری مدل‌ها را به همراه دارد. این پژوهش، پایه‌ای قوی برای ترکیب یادگیری ماشین با مهندسی مبتنی بر مدل ارائه کرده است

هوپنر و همکاران [٤] اندازه و پیچیدگی تبدیل مدل‌ها را در سه زبان مختلف بررسی کردند. آن‌ها از معیارهای پیچیدگی McCabe، تعداد خطوط کد و تعداد کلمات برای سنجش پیچیدگی تبدیل‌ها استفاده کردند. نتایج نشان داد که زبان‌های تبدیل مدل بر کاهش پیچیدگی تأثیر دارند، اما توزیع آن‌ها در کل کد تغییر می‌کند. این پژوهش اهمیت استفاده از زبان‌های سطح بالا مانند جاوا را در کاهش پیچیدگی فرایندهای مدل‌سازی نشان داد.

روحی و همکاران [٥] یک فرامدل عمومی برای تعریف الگوهای طراحی در تبدیل مدل پیشنهاد کردند. آن‌ها با استفاده از مدل‌سازی رسمی و ریاضی، یک زبان یکنواخت و دقیق برای توصیف الگوهای تبدیل مدل توسعه دادند. این رویکرد، بهبود شفافیت و کارایی در فرایندهای تبدیل مدل را به همراه داشته و پیاده‌سازی تبدیل‌های پیچیده را ساده‌تر کرده است.

ویدمان و همکاران [٧] در پژوهش خود به بررسی و توسعه ابزارهای اشکال‌زدایی تبدیل مدل پرداختند و دو ویژگی کلیدی شامل پشتیبانی از نقاط شکست در فرایند اشکال‌زدایی و بهبود اشکال‌زدایی در تبدیل مدل‌های همگام‌سازی را معرفی کردند. نتایج این مطالعه نشان داد که بهبود این ویژگی‌ها منجر به افزایش کیفیت ابزارهای تبدیل مدل شده و کارایی آن‌ها را در تبدیل خودکار مدل‌ها ارتقا می‌دهد.

هوپنر و همکاران [٨] یک مطالعه گسترده مصاحبه‌ای با ٥٦ متخصص از دانشگاه و صنعت انجام دادند تا ویژگی‌های کلیدی زبان‌های تبدیل مدل را شناسایی کنند. نتایج نشان داد که عوامل مهمی مانند قدرت بیان زبان‌های عمومی و قابلیت‌های خاص دامنه‌ای، تأثیر زیادی بر انتخاب زبان‌های تبدیل دارند. این پژوهش درک بهتری از نیازهای واقعی در محیط‌های صنعتی برای انتخاب زبان‌های تبدیل مدل ارائه کرد.

آیزنبرگ و همکاران [١٠] نخستین چارچوب جامع برای استفاده از یادگیری تقویتی در تبدیل مدل درجا[6] (بهینه‌سازی یا اصلاح مدل موجود) را ارائه دادند. این پژوهش نشان داد که الگوریتم‌های یادگیری تقویتی می‌توانند در بهینه‌سازی قوانین تبدیل مدل عملکردی قابل رقابت با روش‌های سنتی داشته باشند. آن‌ها روش‌های مبتنی بر ارزش و مبتنی بر سیاست را برای بهینه‌سازی

6-In-Place Model Transformation

قوانین تبدیل آزمایش کردند و دریافتند که یادگیری تقویتی می‌تواند فرایند تبدیل را انعطاف‌پذیرتر و مقیاس‌پذیرتر کند

لانو و همکاران [۱۲] از یادگیری ماشین نمادین برای کاهش زمان و تلاش مورد نیاز در توسعه تولیدکننده‌های کد استفاده کردند. آن‌ها این روش را در تولید کد از UML به جاوا به‌کار بردند و دریافتند که یادگیری ماشین می‌تواند توسعه تولیدکننده‌های کد را تسریع کند و دقت خروجی را افزایش دهد.

لانو و همکاران [۱۳] یک روش ترکیبی برای تحلیل خودکار نیازمندی‌ها و تطبیق فرامدل ارائه دادند. آن‌ها با استفاده از تکنیک‌های نمادین و تحلیل فرامدل، توانستند کیفیت تبدیل مدل را بهبود دهند. این پژوهش نشان داد که استفاده از روش‌های نمادین در تحلیل و تبدیل مدل می‌تواند فرایندهای مهندسی مدل‌رانده را کارآمدتر کند.

آیزنبرگ و همکاران [۱٤] یک چارچوب یادگیری تقویتی برای بهینه‌سازی تبدیل مدل‌ها پیشنهاد دادند. آن‌ها نشان دادند که یادگیری تقویتی، به‌ویژه در مسائل تک‌هدفه، می‌تواند بر روش‌های مبتنی بر الگوریتم‌های ژنتیکی برتری داشته باشد. نتایج این پژوهش، مزایای ترکیب الگوریتم‌های یادگیری تقویتی با روش‌های کلاسیک مدل‌سازی را برجسته کرده است.

بورگونیو و همکاران [۱۵] یک معماری شبکه عصبی برای تبدیل مدل‌های ناهمگن معرفی کردند. این شبکه، مبتنی بر حافظه کوتاه‌مدت و سازوکار توجه بود که از مثال‌های ورودی- خروجی برای یادگیری قوانین تبدیل مدل استفاده می‌کرد. این پژوهش نشان داد که با استفاده از داده‌های کافی، شبکه‌های عصبی می‌توانند عملیات تبدیل مدل را به‌صورت خودکار انجام دهند.

جدول ۱، خلاصه‌ای از مقالات بررسی شده را ارائه می‌دهد که شامل اطلاعاتی درباره محققان، سال انتشار و میزان استفاده از هوش مصنوعی، یادگیری تقویتی، تبدیل مدل و پشتیبانی از تبدیل مدل با مثال است. در این پژوهش، الگوریتم کیوموتور مبتنی بر یادگیری Q را پیشنهاد می‌کنیم که علاوه بر کاهش وابستگی به داده‌های آموزشی با کیفیت، سازگاری بیشتری با تبدیل‌های ناهمگن دارد. در ادامه، الگوریتم کیوموتور به‌طور دقیق توضیح داده خواهد شد.

پژوهش‌های پیشین در زمینه تبدیل مدل، پیشرفت‌های قابل‌توجهی را در این حوزه رقم زده‌اند، اما همچنان با محدودیت‌هایی روبه‌رو هستند. روش‌های مبتنی بر شبکه‌های عصبی، اگرچه توانایی یادگیری الگوهای پیچیده را دارند، اما به مجموعه داده‌های بزرگ و باکیفیت وابسته هستند که باعث محدود شدن مقیاس‌پذیری و سازگاری آن‌ها می‌شود. از سوی دیگر، رویکردهای سنتی مبتنی بر زبان‌های تبدیل مدل یا چارچوب‌های برنامه‌نویسی، نیازمند مداخله دستی زیادی هستند که آن‌ها را برای تبدیل‌های پیچیده و پویای مدل‌ها ناکارآمد می‌کند.

در مقابل، الگوریتم کیوموتور که بر پایه یادگیری تقویتی توسعه یافته است، سه مزیت کلیدی ارائه می‌دهد. مزیت اول، کاهش وابستگی به داده‌های برچسب‌گذاری‌شده است که این روش برخلاف شبکه‌های عصبی که نیازمند داده‌های آموزشی وسیع هستند، از طریق تعامل مستقیم با محیط یاد می‌گیرد و بدون نیاز به مجموعه داده‌های گسترده، قوانین تبدیل را بهینه می‌کند. مزیت دوم سازگاری پویا با تغییرات مدل‌ها است. الگوریتم یادگیری تقویتی به‌صورت خودکار قوانین تبدیل را اصلاح می‌کند و می‌تواند با ناسازگاری‌ها و ناهمگونی‌های مدل‌های مبدا و مقصد سازگار شود، در حالی که روش‌های دستی چنین انعطافی ندارند. مزیت آخر، بهبود مقیاس‌پذیری است. این روش با تمرکز بر خودکارسازی فرایند نگاشت کلاس‌ها، ویژگی‌ها و روابط، بدون نیاز به تنظیمات خاص دامنه، می‌تواند برای مدل‌های بزرگ نیز بهینه و کارآمد باقی بماند. در مجموع، رویکرد کیوموتور محدودیت‌های روش‌های موجود را کاهش داده و رویکردی هوشمند، انعطاف‌پذیر و مقیاس‌پذیر برای تبدیل مدل‌ها ارائه می‌دهد که می‌تواند در سناریوهای پیچیده‌تر نیز عملکرد مطلوبی داشته باشد.

جدول (۱): مقایسهٔ کارهای مرتبط (تبدیل مدل با مثال (MTBE)، تبدیل مدل (MT)، یادگیری تقویتی (RL)، هوش مصنوعی (AI)، پشتیبانی (✓)، عدم پشتیبانی (×))

معایب	مزایا	AI	RL	MT	MTBE	سال	مرجع
محدودیت در مدل‌سازی چند کاربره پیچیدگی در تنظیم معیارهای کیفیت تهدید به تعمیم‌پذیری نتایج	رویکرد مبتنی بر کیفیت ارزیابی جامع و چندوجهی	✓	✓	×	×	۲۰۲۲	[۱]
وابستگی به کیفیت مثال‌های آموزشی محدود بودن دامنه آزمایش‌ها عدم پوشش رشته‌ها	نوآوری در تعریف معیار تنوع اجتماعی ارائه‌ی چارچوب قابل‌تعمیم برای مسائل مختلف تحلیل آماری قوی و تکرارپذیری بالا	✓	×	×	×	۲۰۲۲	[۲]
محدود بودن به ATL و Java پیچیدگی اجرای پژوهش برای سایر زبان‌ها عدم بررسی عملکرد	تحلیل تطبیقی جامع با داده‌های تجربی استفاده از معیار های چندگانه برای سنجش کیفیت نشان دادن تاثیر نسخه‌های جدید Java	×	×	✓	×	۲۰۲۱	[۴]
تعداد محدود الگوهای بررسی شده نبود ابزار کاربردی برای کاربران نهایی بررسی تجربی محدود	ارائه یک صوری‌سازی جامع برای الگوهای تبدیل قابلیت تعمیم به زبان‌های مختلف تبدیل مدل کاربردپذیری بر روی الگوهای واقعی و پراستفاده	×	×	✓	×	۲۰۲۲	[۵]
نبود ارزیابی کمی و تجربی دقیق عدم پشتیبانی از دیباگ پیاده‌سازی ناقص برخی قابلیت‌ها	افزایش تعامل کاربر با فرآیند تبدیل مدل‌ها رابط کاربری بصری و قابل پیکربندی پشتیبانی از هم‌زمان‌سازی مدل‌ها	×	×	✓	×	۲۰۲۲	[۷]
مشکل در اشکال‌زدایی و آزمون پشتیبانی و اکوسیستم محدودتر منحنی یادگیری بالا	افزایش بهره‌وری توسعه‌دهنده پشتیبانی از مفهوم سطح بالا پشتیبانی از سازگاری با فرامدل‌ها	×	×	✓	×	۲۰۲۲	[۸]
مقیاس‌پذیری پایین الگوریتم‌ها وابستگی شدید به طراحی تابع نبود پشتیبانی از بهینه‌سازی چندهدفه	ارزیابی عملی روی چندین مطالعه موردی واقعی مقایسه با الگوریتم NSGA-II	✓	✓	✓	×	۲۰۲۱	[۱۰]
نیاز به طراحی راهبرد تبدیل پیش از شروع نیاز به دسته‌بندی دستی مثال‌ها وابستگی به درخت نحوی زبان‌ها	کاهش قابل توجه تلاش انسانی عدم نیاز به داده‌های آموزشی بزرگ دقت بالا در ارزیابی	✓	×	✓	×	۲۰۲۲	[۱۲]
محدودیت در پوشش کامل قواعد پیچیده نیاز به داده‌های نمونه کافی عدم مقایسه دقیق با سایر ابزارهای مطرح	کاربرد عملی و پیاده‌سازی واقعی کاهش وابستگی به متخصصان تبدیل مدل ایجاد قواعد تبدیل قابل ویرایش	×	×	✓	✓	۲۰۲۱	[۱۳]
مقیاس‌پذیری محدود در مسائل بزرگ پیچیدگی تنظیم پارامترها وابستگی به ساختار Henshin	گسترش ابزار MOMoT پوشش هر دو حالت تک‌هدفه و چندهدفه	✓	✓	✓	×	۲۰۲۴	[۱۴]
نیاز به داده‌های آموزشی زیاد محدودیت در پردازش ورودی‌های بزرگ ناتوانی در انجام عملیات عددی یا منطقی وابستگی بالا به کیفیت داده	نوآوری در استفاده از LSTM در مهندسی مدل پوشش تبدیل‌های ناهمگون ساختار معماری قابل تکرار و عمومی مدیریت واژگان نامحدود	✓	×	✓	×	۲۰۲۲	[۱۵]
وابستگی به ساختار گراف و استخراج ویژگی‌ها عدم استفاده از یادگیری عمیق برای نگاشت‌های انتزاعی ناتوانی در پردازش مدل‌های بسیار پیچیده	کاهش وابستگی به داده‌های آموزشی افزایش مقیاس‌پذیری و کارایی پشتیبانی از نگاشت‌های ترکیبی برای ویژگی‌ها پوشش دامنه‌ای بالا در مطالعات موردی استفاده از معیارهای ارزیابی دقیق	✓	✓	✓	✓	۲۰۲۵	کیوموتور

۳- کیوموتور: الگوریتم پیشـــنهادی مبتنی بر یادگیری تقویتی

در این پژوهـش، الگوریتم کیوموتور روشی مبتنی بر یادگیری تقویتی (یادگیری Q) برای تبدیل مدل‌ها ارائه شده است. این الگوریتم توسـعه یافتهٔ الگوریتم[۱۶] می‌باشد و مبتنی بر فرهنگ لغات کلید- مقدار اسـت. کیوموتور برای خودکارسازی تبدیل مدل شامل سه مرحله اصلی است که در آن فرامدل‌های مبدا و مقصد تحلیل شـده، ویژگی‌ها و روابط استخراج می‌شوند و سپس فرایند یادگیری تقویتی برای تعیین نگاشت‌های بهینه اجرا می‌شود.

در مرحله نخسـت، داده‌های ورودی آماده‌سازی شده و اطلاعات سـاختاری مدل‌ها اسـتخراج می‌شـود. برای تسـهیل پـردازش، فرامدل‌های مبدا و مقصد به سـاختار کلید- مقدار تبدیل می‌شـوند تا عناصـر مدل‌ها به‌صورت داده‌های سـاختاریافته نمایش داده شـوند. علاوه بر این، مدل‌های مبـدا و مقصد در قالب نمودارهای گرافی نمایش داده می‌شوند، به‌طوری که گره‌ها نشان‌دهنده عناصر مدل و یال‌ها نشـان‌دهنده روابط بین آن‌ها هستند. این نمایش، امکان مقایسـه دقیق‌تر عناصر مدل‌های مختلف را فراهم می‌کند.

در مرحلـه دوم، ویژگی‌ها و روابط مدل‌ها اسـتخراج و پردازش می‌شـوند. کلاس‌ها و موجودیت‌های مدل‌های مبدا و مقصد شناسایی شده و ویژگی‌های آن‌ها به‌صورت سـه‌تایی‌های استاندارد شـامل «نام کلاس، نام ویژگی و نوع ویژگی» اسـتخراج می‌شـوند. همچنیـن، روابط میان عناصر مدل مشـخص شـده و به‌صورت گراف جهت‌دار ارائه می‌شوند که در آن، ارتباط بین کلاس‌ها و ویژگی‌های متناظر در هر دو مدل مورد بررسی قرار می‌گیرد.

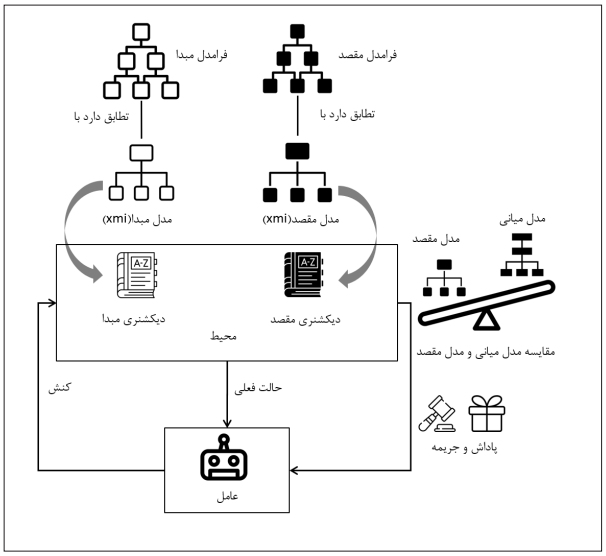
در مرحله سـوم، فرایند یادگیری نگاشـت‌های مدل با اسـتفاده از یادگیری Q اجرا می‌شود. در این مرحله، عامل یادگیری تقویتی با محیط تعامل کرده و بر اساس ساختار مدل‌هـای مبدا و مقصد، قوانین تبدیـل را یاد می‌گیرد. این فرایند شـامل مجموعه‌ای از عملیـات تبدیل مانند افزودن، حذف یا اصلاح عناصر و روابط است. عامل یادگیری، پس از هر اقدام، میزان موفقیت خود را بر اسـاس میزان تطابق مدل تبدیل‌شده با مدل مقصد ارزیابی کرده و متناسب با آن پاداش دریافت می‌کند. این پاداش به‌روزرسانی جدول Q را امکان‌پذیر ساخته و موجب بهبود تدریجی فرایند تطبیق و افزایش دقت تبدیل می‌شود.

با اجرای این سـه مرحله، الگوریتم کیوموتور می‌تواند نگاشـت‌های بهینه بین مدل‌های مبـدا و مقصد را به‌طور خودکار تعیین کند و نیاز به تعریف دستی قوانین تبدیل را کاهش دهد، در حالی که دقت، انعطاف‌پذیری و مقیاس‌پذیری فرایند تبدیل را افزایش می‌دهد.

برای پیاده‌سـازی کیوموتـور، از کتابخانه PyEcore جهت پردازش فرامدل‌های Ecore اسـتفاده شـده اسـت. مدل‌های مبدا و مقصد در قالب XML نمایش داده شده‌اند و ساختار آن‌ها به‌صورت گراف‌های جهت‌دار برای پردازش و تحلیل اسـتفاده شده است. پس از پیاده‌سازی، الگوریتم روی چندین مطالعه موردی اسـتاندارد اجرا شد تا میزان دقت، صحت و کارایی آن ارزیابی شود.

پس از استخراج اطلاعات از فرامدل‌ها، مدل‌های مبدا و مقصد که در قالب XML نمایش داده شده‌اند، به گراف‌های جهت‌دار تبدیل می‌شوند تا امکان تحلیل و تجسم ساختاری آن‌ها فراهم شـود. در این فراینـد، عناصر XML تجزیه و پردازش شده و هر کلاس به‌عنوان یک گره در گراف نمایش داده می‌شـود، در حالی که روابط بیـن کلاس‌ها از طریق یال‌های جهت‌دار تعریف می‌شوند که سلسله‌مراتب عناصر مدل را منعکس می‌کنند. برچسـب یال‌ها نشـان‌دهنده نام کلاس‌ها و ویژگی‌های آن‌ها اسـت. در نهایت، این گراف‌ها به‌صورت بصری نمایش داده شـده و امکان مقایسـه و اعتبارسـنجی مدل‌های مبدا و مقصد را برای فرایند تبدیل مدل فراهم می‌کنند.

فرایند ارائه شده در شکل ۱، الگوریتم کیوموتور را برای تبدیل مدل با استفاده از یادگیری تقویتی نشان می‌دهد.



شکل (۱): فرایند آماده‌سازی و استفاده از الگوریتم کیوموتور

۳-۱- الگوریتم کیوموتور: روشی توسعه‌پذیر برای نگاشت مدل‌ها

شبه‌کد الگوریتم کیوموتور در شکل ۲ نمایش داده شده است. الگوریتم پیشنهادی بر پایه یادگیری تقویتی و مبتنی بر جدول Q طراحی شده است تا فرایند نگاشت مدل‌ها را بهینه‌سازی کند. هدف این الگوریتم، یادگیری بهترین راهکار برای تبدیل یک مدل مبدا به مدل مقصد با استفاده از اطلاعات فرامدل‌های مبدا و مقصد است. در این راستا، الگوریتم تلاش می‌کند تا مطلوب‌ترین نگاشت را میان عناصر دو مدل تعیین نماید.

در مرحلهٔ اول، مقداردهی اولیه انجام می‌شود. در این مرحله، یک جدول Q به‌عنوان ساختار ذخیره‌سازی مقادیر Q مقداردهی شده و محیط شامل فرامدل‌های مبدا و مقصد تنظیم می‌شود تا چارچوب تصمیم‌گیری عامل یادگیری تقویتی مشخص گردد.

در ادامه، فرایند یادگیری و نگاشت مدل‌ها اجرا می‌شود. الگوریتم در یک حلقه اصلی تکرار می‌شود که تعداد دفعات اجرای آن بر اساس مقدار episodes تعیین شده است. در ابتدای هر تکرار، حالت اولیه تنظیم شده و فرایند یادگیری تا رسیدن به یک نگاشت کامل و معتبر میان مدل‌های مبدا و مقصد ادامه پیدا می‌کند.

در هر گام، مجموعه‌ای از نگاشت‌های بالقوه برای تطبیق عناصر مدل مبدا با مدل مقصد ایجاد می‌شود. به‌منظور کاهش فضای جستجو و جلوگیری از افزایش نمایی حالات، برخی از سیاست‌های خاص با احتمال بیشتری در نظر گرفته می‌شود. به‌عنوان مثال، در نگاشت کلاس به کلاس، کلاس‌هایی که تعداد نمونه‌های برابری در مدل‌های مبدا و مقصد دارند، احتمال انتخاب بالاتری دارند. در نگاشت ویژگی به ویژگی، مقادیر و تعداد نمونه‌ها مقایسه شده و نمونه‌هایی که یکسان هستند، احتمال بالاتری در نگاشت پیدا می‌کنند. همچنین، در نگاشت رابطه به رابطه، بررسی می‌شود که تعداد اشیای کلاس‌های مرتبط در هر دو مدل یکسان باشد.

```
Require:
initial_state, SMM (source_metamodel), TMM (target_metamodel), SM (source_model), TM (target_model), episodes, α, γ, ϵ, min_ϵ, ϵ_decay
Initialization:
q_table ← empty dictionary
env ← (initial_state, SMM, TMM, SM, TM)
for episode = 1 to episodes do
    state ← copy of initial_state
    while ¬ is_terminal_state(state, TMM) do
        actions ← generate_possible_actions(state, SMM, TMM, SM, TM)
        if actions is empty then
            break
        end if
        if random value < ϵ then
            action ← random action from actions
        else
            q_vals ← { a: q_table.get((state, a), 0) for a in actions }
            action ← argmax(q_vals) if q_vals else random action
        end if
        new_state ← apply_action(state, action)
        if action is class_mapping then
            reward ← cal_class_reward(state, new_state, SM, TM)
        else if action is attribute_mapping then
            reward ← cal_attr_reward(state, new_state, SM, TM, SMM, TMM)
        else if action is reference_mapping then
            reward ← cal_ref_reward(state, new_state, SM, TM, SMM, TMM)
        end if
        state ← new_state
        nxt_acts ← generate_possible_actions(state, SMM, TMM, SM, TM)
        if nxt_acts is not empty then
            nxt_max_vals ← [q_table.get((state, a), 0) for a in nxt_acts]
            nxt_max ← max of nxt_max_vals
        else
            nxt_max ← 0
        end if
        old_q ← q_table.get((state, action), 0)
        q_table[(state, action)] ←
        old_q + α × (reward + γ × nxt_max - old_q)
    end while
    ϵ ← max(min_ϵ, ϵ × ϵ_decay)
end for
return q_table, state
```

شکل (۲): شبه‌کد الگوریتم کیوموتور

پس از تولید نگاشت‌های ممکن، بهترین نگاشت انتخاب و اعمال می‌شود. اگر هیچ عمل معتبری در یک گام قابل اجرا نباشد، حلقه یادگیری متوقف می‌شود. در غیر این صورت، یک عمل از میان دو راهکار اکتشاف[7] یا بهره‌برداری[8] انتخاب شده و اعمال می‌شود. پس از اعمال نگاشت، مدل مبدا به مدل میانی جدیدی تبدیل می‌شود و میزان پاداش مربوط به نگاشت انجام شده محاسبه می‌گردد. تفکیک پاداش‌ها بر اساس نوع نگاشت صورت گرفته تا از تأثیر نامناسب یک نوع نگاشت بر سایر انواع جلوگیری شود.

تابع پاداش در الگوریتم کیوموتور به‌منظور ارزیابی میزان انطباق ساختاری مدل تبدیل‌شده با مدل هدف به‌صورت فرمول (۱) تعریف می‌شود:

$$R = \begin{matrix} \min(N_t, N_g) \times W_s + \\ \left(|E_g - E_t| - E_c\right) \times W_e + \\ \left(|N_g - N_t| \times W_n\right) \end{matrix} \qquad (۱)$$

که در آن، R مقدار پاداش نهایی، N_t تعداد گره‌ها در مدل تبدیل‌شده، N_g تعداد گره‌ها در مدل هدف، N_c تعداد گره‌هایی که به درستی تبدیل شده‌اند (گره‌های مشترک بین مدل هدف و تبدیل‌شده) می‌باشد. همچنین، E_c تعداد یال‌ها در مدل تبدیل‌شده، E_g تعداد یال‌ها در مدل هدف، E_c تعداد یال‌هایی که به درستی تبدیل شده‌اند (یال‌های مشترک بین مدل هدف و تبدیل‌شده) بوده و اوزان به‌صورت W_n وزن اختصاص‌یافته برای پاداش گره‌ها، W_e وزن اختصاص‌یافته برای پاداش یال‌ها و Ws وزن اختصاص‌یافته برای شباهت ساختاری کلی هستند که در شبه‌کد به‌ترتیب ۵، ۲ و ۵ مقداردهی شده‌اند.

در چارچوب الگوریتم کیوموتور فضای حالت شامل وضعیت‌های ممکن مدل در طول فرآیند تبدیل است که به‌صورت گرافی بازنمایی می‌شوند. هر حالت نمایانگر یک گام میانی در تبدیل مدل مبدأ به مدل مقصد است، که الگوریتم باید تصمیم بگیرد کدام عنصر از مدل مبدأ برای نگاشت به یک عنصر خاص از مدل مقصد مناسب است. فضای کنش نیز مجموعه‌ای از اعمال مجاز را شامل می‌شود که الگوریتم می‌تواند برای بهبود مدل تبدیل‌شده به کار گیرد؛ این اعمال شامل نگاشت مستقیم، ترکیب ویژگی‌ها و عملیات افزودن، حذف یا اصلاح بر روی گره‌ها و یال‌های گراف مدل می‌باشد. این ساختار تعریف‌شده برای حالت و کنش، امکان یادگیری تدریجی بهترین مسیر تبدیل را از طریق به‌روزرسانی‌های متوالی تابع Q فراهم می‌کند.

برای کاهش فضای جستجو و هدایت مؤثرتر فرآیند یادگیری، برخی سیاست‌های خاص به‌عنوان شرط اولیه برای تعریف فضای حالت در نظر گرفته شده‌اند. به‌عنوان مثال، ابتدا نگاشت‌هایی بررسی می‌شوند که کلاس‌های مبدا و مقصد دارای تعداد نمونهٔ مشابه باشند یا ویژگی‌هایی با مقادیر یکسان (یا بسیار مشابه) داشته باشند. این سیاست‌ها به عنوان فیلتر اولیه پیش از اجرای الگوریتم اعمال می‌شوند تا از یادگیری روی نگاشت‌های غیرمنطقی جلوگیری شود و نرخ همگرایی بهبود یابد.

پس از دریافت پاداش، جدول Q طبق معادله بلمن به‌روزرسانی می‌شود تا راهبرد تصمیم‌گیری عامل یادگیری بهبود یابد. مقدار احتمال اکتشاف (ε) نیز به‌تدریج کاهش می‌یابد تا الگوریتم از فاز اکتشاف به سمت بهره‌برداری حرکت کند و تصمیم‌گیری‌های بهینه‌تر را در پیش بگیرد.

در نهایت، جدول Q نهایی شده و وضعیت نهایی به‌عنوان خروجی ارائه می‌شود. با کاهش مقدار ε، عامل یادگیری به‌جای انتخاب‌های تصادفی، از دانش آموخته‌شده برای تصمیم‌گیری بهره می‌گیرد. این امر موجب بهبود دقت و اطمینان در فرایند نگاشت مدل‌ها شده و کیفیت تبدیل مدل‌ها را افزایش می‌دهد.

در الگوریتم کیوموتور، پارامتر اپیزودها حداکثر تعداد تکرارهای فرآیند را مشخص می‌کند، اما امکان توقف زودهنگام نیز وجود دارد. در هر اپیزود، گراف مدل میانی با گراف مدل مقصد مقایسه می‌شود و در صورت تطابق کامل این دو گراف، الگوریتم متوقف می‌شود. نگاشت کامل و معتبر زمانی حاصل می‌شود که گراف مدل میانی

7-Exploration
8-Exploitation

از نظر ساختاری و معنایی با گراف مدل مقصد یکسان باشد. این تطابق از طریق بررسی ویژگی‌های گراف‌ها در پایان هر اپیزود تأیید می‌شود. جدول Q، که در طول اجرا به‌روزرسانی می‌شود، قوانین تبدیل را ذخیره کرده و به‌عنوان خروجی الگوریتم ارائه می‌شود.

۳-۲- معیارهای ارزیابی الگوریتم کیوموتور

معیارهای مقایسهٔ مدلی که مورد بررسی قرار گرفته‌اند عبارتند از: دقت[۹]، حساسیت[۱۰] یا بازیابی[۱۱]، امتیاز F1[۱۲] و صحت[۱۳]. این معیارها به چهار مقدار مثبت‌های درست[۱۴]، مثبت‌های کاذب[۱۵]، منفی‌های درست[۱۶] و منفی‌های کاذب[۱۷] وابسته هستند.

مثبت‌های درست (TP): تعداد گره‌ها یا یال‌هایی که در هر دو گراف مقصد و میانی (گرافی که در طول اجرا ساخته می‌شود) وجود دارند. به‌عبارت دیگر، تطبیق درست اندازه‌گیری می‌شود یعنی پیش‌بینی شده که یک گره یا یال باید وجود داشته باشد و این پیش‌بینی صحیح بوده است.

مثبت‌های کاذب (FP): تعداد گره‌ها یا یال‌هایی که در گراف میانی وجود دارند ولی در گراف هدف وجود ندارند. یعنی پیش‌بینی شده که یک گره یا یال باید وجود داشته باشد، ولی این پیش‌بینی اشتباه بوده است.

منفی‌های درست (TN): تعداد گره‌ها یا یال‌هایی که در هیچ‌کدام از گراف‌ها وجود ندارند. در گراف‌ها محاسبه منفی‌های درست معمولاً سخت‌تر است، زیرا شامل فضای کل گره‌ها و یال‌های ممکن است.

منفی‌های کاذب (FN): تعداد گره‌ها یا یال‌هایی که در گراف هدف وجود دارند ولی در گراف میانی وجود ندارند. یعنی پیش‌بینی شده که یک گره یا یال نباید وجود داشته باشد، ولی در گراف هدف وجود دارد.

این چهار اصطلاح در ماتریسی به نام ماتریس درهم‌ریختگی[۱۸] خلاصه می‌شوند. این ماتریس در جدول ۲ قرار داده شده است.

جدول (۲): ماتریس درهم ریختگی

کل	منفی پیش‌بینی	مثبت پیش‌بینی	جمع واقعی
مثبت واقعی	منفی‌های کاذب	مثبت‌های درست	مثبت واقعی
منفی واقعی	منفی‌های درست	مثبت‌های کاذب	منفی واقعی
جمع پیش‌بینی	منفی پیش‌بینی شده	مثبت پیش‌بینی شده	کل نمونه‌ها

در ادامه فرمول‌های مربوط به معیارهای ارزیابی بر اساس مقادیر معرفی شده، آمده است. فرمول ۲ معیار دقت را نشان می‌دهد. این معیار نشان می‌دهد چند درصد از پیش‌بینی‌های مثبت، واقعاً مثبت بودند.

$$Precision = \frac{TP}{TP + FP} \qquad (۲)$$

فرمول ۳ معیار حساسیت یا recall را نشان می‌دهد. این معیار نشان می‌دهد چند درصد از موارد مثبت واقعی توسط مدل شناسایی شدند.

$$Recall = \frac{TP}{P} \qquad (۳)$$

فرمول ٤ معیار صحت را نشان می‌دهد. این معیار نشان می‌دهد مدل چه درصدی از کل نمونه‌ها را به‌درستی طبقه‌بندی کرده است.

$$Accuracy = \frac{TP + TN}{P + N} \qquad (٤)$$

فرمول ٥ معیار F1-Score را نشان می‌دهد. این معیار میانگین هارمونیک بین precision و recall است و تعادلی بین این دو معیار ایجاد می‌کند.

$$\mathrm{F1-score} = \frac{2 * Precision * Recall}{Precision + Recall} \qquad (٥)$$

فرمول ٦ معیار ضریب همبستگی متیوز[۱۹] را نشان می‌دهد. MCC شاخصی برای ارزیابی عملکرد مدل‌های طبقه‌بندی دوتایی است که دقت کلی مدل را حتی در شرایط

9-Precision
10-Sensitivity
11-Recall
12-F1-Score
13-Accuracy
14-True Positives
15-False Positives
16-True Negatives
17-False Negatives

18-Confusion Matrix
19-Matthews Correlation Coefficient-MCC

داده‌های نامتوازن نشـــان می‌دهد. مقدار آن بین ۱- تا ۱+ است که عدد بالاتر نشان‌دهنده عملکرد بهتر است.

$$MCC = \frac{TP.TN - FP.FN}{\sqrt{(TP+FP).(TP+FN).(TN+FP).(TN+FN)}} \quad (٦)$$

۳-۳- مثال تشریحی از فرایند تبدیل

بـــرای درک بهتر الگوریتم مثـــالی در این بخش آورده شده است. این مثال تبدیل نمودار کلاس به شمای رابطه‌ای می‌باشد.

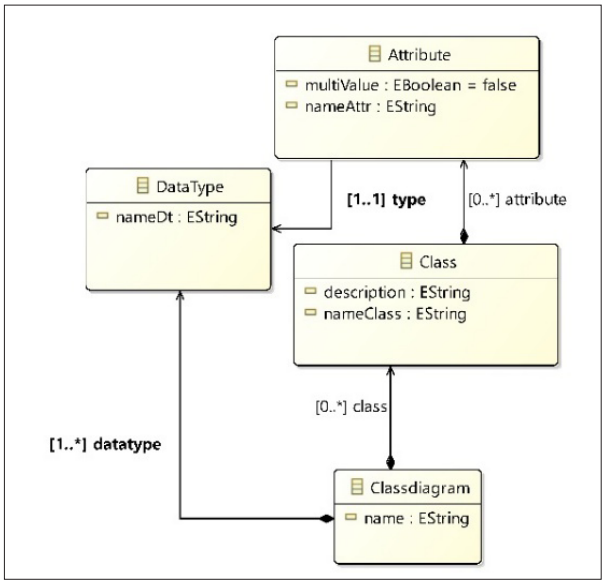
عناصـــر اصـــلی فرامـــدل مبـــدا شـــامل فـــراکلاس Classdiagram ریشـــه فرا مدل است و شامل مجموعه‌ای از کلاس‌ها و انواع داده است و با استفاده از صفت name، نام نمودار کلاس مشخص می‌شود.

فراکلاس Class نمایندهٔ یک کلاس در فرا مدل است که دارای صفت‌هـــای nameClass: EString برای نام کلاس و description: EString بـــرای توضیحات مربوط به کلاس است و می‌تواند شامل چندین صفت باشد.

فـــراکلاس Attribute نمایندهٔ یـــک صفت مرتبط با یک کلاس است که با صفت‌های multiValue: EBoolean برای تعیین چندارزشی یا تکارزشی بودن و nameAttr: EString برای نام صفت مشخص می‌شود و هر صفت متعلق به یک کلاس با نوع داده خاصی است.

DataType نمایندهٔ انـــواع داده مانند Integer و String در فرامدل اســـت که با صفت nameDt: EString برای نام نوع داده مشخص می‌شود.

روابط در فرامدل شـــامل موارد زیر اســـت: رابطه بین Classdiagram و Class به‌صـــورت [*..۰] class نشـــان می‌دهد که هر نمـــودار کلاس می‌تواند صفر یا چند کلاس داشته باشد؛ رابطه بین Class و Attribute به‌صورت [*..۰] attribute نشان می‌دهد که هر کلاس می‌تواند صفر یا چند ویژگی داشـــته باشـــد؛ رابطه بین Attribute و DataType به‌صورت [۱..۱] type مشـــخص می‌کند که هر ویژگی باید



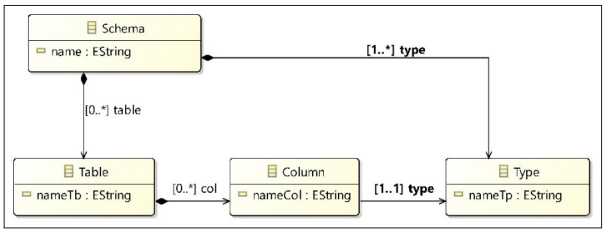
شکل (۳): فرامدل مبدا: نمودار کلاس

دقیقاً یک نوع داده داشـــته باشد؛ رابطه بین Classdiagram و DataType به‌صورت [۱..*] datatype نشان می‌دهد که هر نمودار کلاس باید حداقل یک نوع داده داشته باشد. در شـــکل ۳ فرا مدل مبدا برای تبدیل نمودار کلاس به شمای رابطه‌ای نشان داده شده است.

شـــکل ٤، فرامدل مقصد را که شمای رابطه‌ای می‌باشد نمایـــش می‌دهـــد. عناصر اصـــلی فرا مدل مقصد شـــامل Schema به‌عنوان نماینـــده طرح کلی پایگاه داده با ویژگی name: EString بـــرای نام شِـــمای پایگاه داده اســـت که مجموعـــه‌ای از جداول را در بر می‌گیـــرد؛ Table به‌عنوان نماینده یک جدول با ویـــژگی nameTb: Estring برای نام جدول که شـــامل مجموعه‌ای از ســـتون‌ها است؛ Column به‌عنوان نماینده یک ســـتون با ویژگی nameCol: EString برای نام ســـتون که هر ســـتون دقیقاً یک نـــوع داده دارد؛ و Type به‌عنـــوان نماینده نـــوع داده ســـتون‌ها با ویژگی nameTp: EString برای نام نوع داده است.

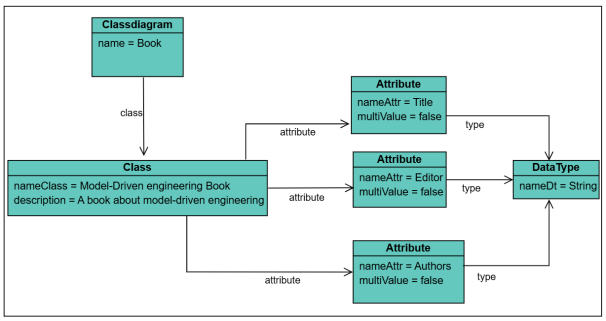
روابط در فرامدل مقصد شامل موارد زیر است: رابطه بین Schema و Table به‌صورت [*..۰] table نشان می‌دهد که هر شِمای پایگاه داده می‌تواند شامل صفر یا چند جدول باشـــد؛ رابطه بیـــن Table و Column به‌صورت [*..۰] col نشـــان می‌دهد که هر جدول می‌تواند شـــامل صفر یا چند

ستون باشد؛ رابطه بین Column و Type به‌صورت [۱..۱] type مشخص می‌کند که هر ستون باید دقیقاً یک نوع داده داشته باشد؛ و رابطه بین Schema و Type به‌صورت [۱..*] type نشان می‌دهد که هر شِمای پایگاه داده شامل یک یا چند نوع داده است. این فرامدل برای مدل‌سازی سیستم‌های مبتنی بر مدل رابطه‌ای ۲۰ طراحی شده است. این فرامدل کاربردهای گسترده‌ای در حوزه‌های مختلف دارد.



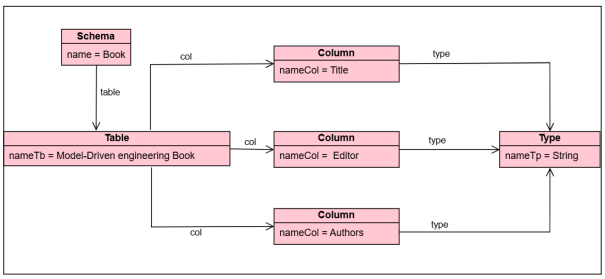
شکل (۴): فرامدل مقصد: شمای رابطه‌ای

برای هر یک از این فرامدل‌ها، نمونه مدل‌هایی طراحی و پیاده‌سازی شده است. این مدل‌ها در قالب فایل‌های XMI، که نوعی قالب مبتنی بر XML در چارچوب مدل‌سازی اکلیپس هستند، تعریف شده‌اند. این فایل‌ها به‌عنوان ورودی الگوریتم پردازش شده و از آن‌ها برای استخراج ساختار کلید–مقدار و گراف مرتبط استفاده می‌شود تا سازگاری لازم با زبان Python فراهم گردد. نمونه‌ای از مدل مبدا مبتنی بر فرامدل نمودار کلاس در شکل ۵ آورده شده است



شکل (۵): مدل مبدا مبتنی فرامدل نمودار کلاس

نمونه‌ای از مدل مقصد مبتنی بر فرامدل شمای رابطه‌ای در شکل ٦ آورده شده است.



شکل (۶): مدل مقصد مبتنی فرامدل شمای رابطه‌ای

در این مرحله تمام قوانین تبدیل برای این مثال در جدول ۳ آورده شده است:

قوانین تبدیل مدل به‌طور خلاصه فرایند نگاشت بین مدل مبدا و مدل مقصد را مشخص می‌کنند.

قانون R1 هر کلاس از نوع Class را به یک جدول از نوع Table تبدیل می‌کند. ویژگی nameTb در جدول جدید برابر با nameClass کلاس اصلی تنظیم می‌شود. همچنین برای هر ویژگی از نوع Attribute در کلاس، یک ستون از نوع Column ساخته می‌شود که نام آن برابر با nameAttr ویژگی اصلی است. ارتباط بین جدول و ستون‌ها با استفاده از مرجع col برقرار می‌شود.

قانون R2 هر نوع داده از نوع DataType را به یک نوع داده در مدل رابطه‌ای از نوع Type تبدیل می‌کند. ویژگی nameAttr در نوع داده جدید برابر با nameDt نوع داده اصلی تنظیم می‌شود.

قانون R3 برای ویژگی‌هایی از نوع Attribute که چند مقداری نیستند یک ستون از نوع Column می‌سازد. نام ستون برابر با nameAttr ویژگی اصلی است و ارتباطاتی برای تعیین مالک ستون ([Table] ← owner) و نوع داده آن ([Type] ← type) برقرار می‌شود.

قانون R4 هر نمودار کلاس از نوع ClassDiagram را به یک شِمای پایگاه داده از نوع Schema تبدیل می‌کند. نام شِمای جدید برابر با name نمودار کلاس اصلی تنظیم می‌شود. سپس برای هر کلاس در نمودار، قانون

20-Relational Model

جدول (۳): قوانین تبدیل کلاس به شمای رابطه‌ای

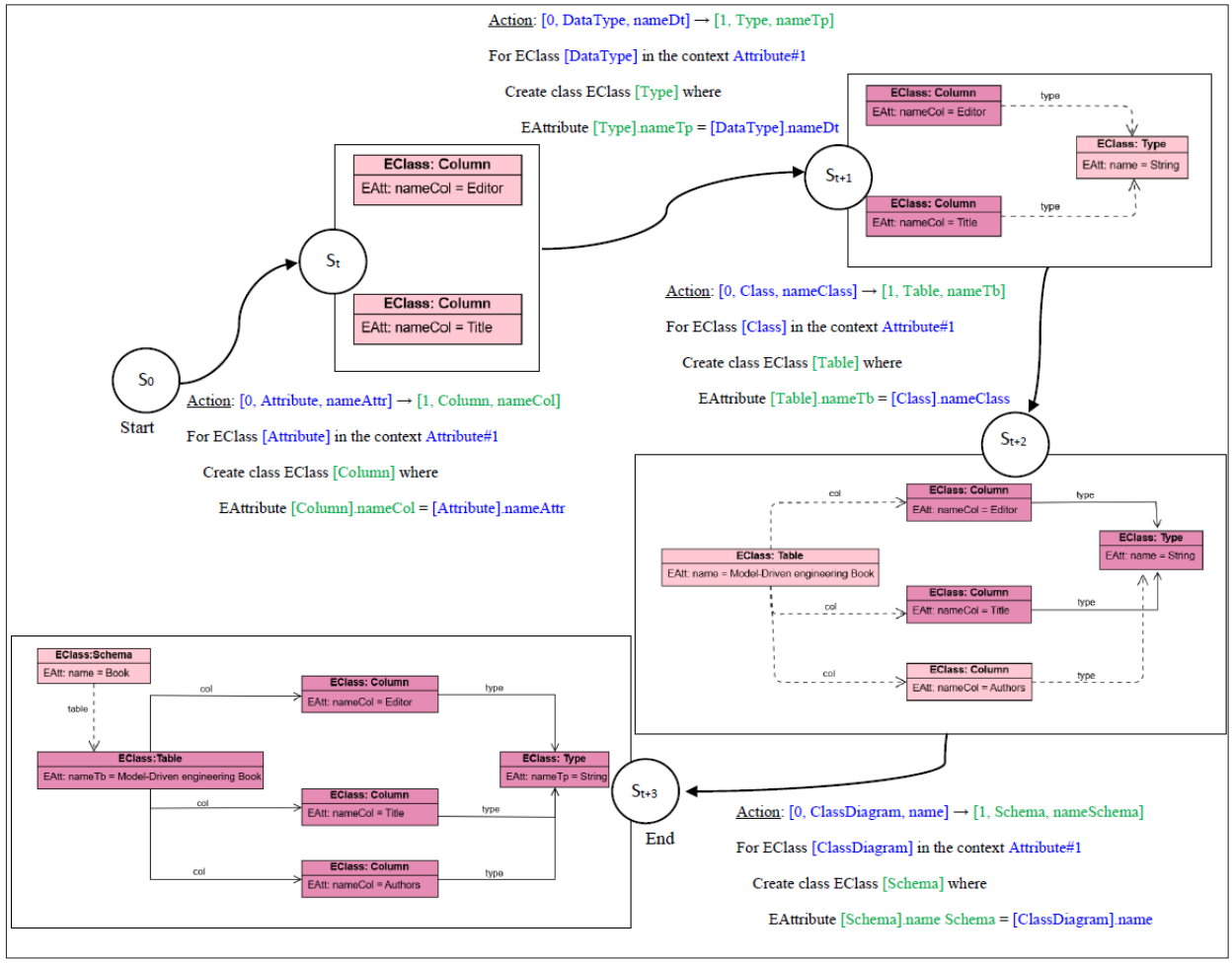
قانون	شماره‌ی قانون
For each class of type [Class] Construct a class of type [Table] with Attribute [Table].nameTb = [Class].nameClass Construct a class of type [Column] with Attribute [Column].nameCol = [Attribute].nameAttr Set reference bindings [Table] → col → [Column]	R1 -> Class → Table
For each class of type [DataType]: Construct a class of type [Type] with Attribute [Type].nameAttr = [DataType].nameDt	R2 -> DataType → Type
For each class of type [Attribute].multiValue = False Construct a class of type [Column] with Attribute [Column].nameCol = [Attribute].nameAttr Set reference bindings [Column] - owner → [Table] Set reference bindings [Column] - type → [Type]	R3 -> Attribute → Column
For each class of type [ClassDiagram] Construct a class of type [Schema] with Attribute [Schema].name = [ClassDiagram].name For each class of type [Class] in [ClassDiagram] Apply Rule R1 (Class → Table) Set reference bindings [Schema] - table → [Table]	R4 -> Classdiagram → Schema

R1 (Class → Table) اعمال می‌شـــود. همچنین ارتباط بین شِـــمای جدید و جداول آن از طریق مرجـــع table برقرار می‌گردد.

در شـــکل ۷ مراحل کار این الگوریتم قرار داده شـــده است. این فرایند به طور خاص برای تبدیل Class Diagram به یک ساختار داده‌ای جدید به نام Schema مورد استفاده قرار می‌گیـــرد. در این تبدیل، ابتدا فرایند از حالت شـــروع آغاز می‌شـــود. ســـپس، برای هر صفت یـــک کلاس جدید به نام «Column» ایجاد می‌شـــود که شـــامل ویژگی‌هایی مانند «nameCol» (برای نگهداری نام ســـتون)، «type» و «nameColEditor» اســـت. همچنین، بـــرای هر نوع داده‌ای (DataType) یـــک کلاس جدیـــد بـــه نام «Type» ســـاخته می‌شـــود که ویژگی‌های مشـــابهی از جمله «nameTp» را برای نام نوع داده‌ای اصلی در بر دارد.

علاوه بـــر این، بـــرای هـــر کلاس (Class) در متن، یک جدول (Table) جدید ایجاد می‌شـــود که نام آن در ویژگی «nameTb» ذخیره می‌شـــود. در نهایـــت، برای هر نمودار کلاس (ClassDiagram) یک شِـــما (Schema) ایجاد شده و نام آن در ویژگی «nameSchema» نگهداری می‌شود. این نمودار کلی نشـــان می‌دهد که چگونه داده‌های موجود در یک نمودار کلاس (شـــامل انواع داده‌ای، صفات و کلاس‌ها) به یک ســـاختار رابطه‌ای جدید تبدیل می‌شوند که می‌توان از آن برای ذخیره داده‌ها در پایگاه داده یا تولید کد در یک زبان برنامه‌نویسی خاص استفاده کرد.

نکتهٔ حائز اهمیت این است که الگوریتم کیوموتور برای نگاشـــت‌های چند به یک برای ویژگیهـــا عمل می‌کند. برای شناسایی خودکار نگاشت‌های ترکیبی در سطح ویژگی‌ها، روشی مبتنی بر مقایسه داده‌های نمونه طراحی شد. در این روش، مقادیر ویژگی‌های مدل مقصد با ترکیب مقادیر دو یا چند ویژگی در مدل مبدأ مقایسه می‌شود و در صورتی



شکل (۷): فرایند تصمیم‌گیری مارکوف برای تبدیل این مثال

که این ترکیب با مقدار ویژگی مقصد مطابقت داشته باشد، به‌عنوان یک نگاشت ترکیبی معتبر شناسایی می‌گردد. این رویکرد علاوه بر کاهش وابستگی به نام ویژگی‌ها، دقت نگاشت‌ها را در سناریوهای واقعی افزایش می‌دهد.

۴- ارزیابی

در این بخش، ارزیابی الگوریتم کیوموتور بر اساس مطالعات موردی و معیارهای مختلف انجام می‌گیرد. سپس، نتایج ارزیابی مطالعه‌های موردی بررسی می‌شود.

۴-۱- معرفی مطالعات موردی

برای ارزیابی عملکرد الگوریتم پیشنهادی، ۸ مطالعه موردی استاندارد و پرکاربرد در حوزه تبدیل مدل انتخاب شده‌اند که به‌عنوان سنجه در مقالات معتبر مورد استفاده قرار گرفته‌اند. این مطالعات شامل تبدیل‌های متنوعی می‌باشند که از مقالات [۱۷ و ۱۸] استخراج شده‌اند. تنوع ساختاری و مفهومی این سنجه‌ها امکان بررسی دقت، انعطاف‌پذیری و مقیاس‌پذیری الگوریتم کیوموتور را در سناریوهای مختلف فراهم می‌کند.

۴-۱-۱- تبدیل انتشارات به کتابخانه

در این مطالعه موردی [۱۷]، فرایند تبدیل مدل انتشارات به مدل کتابخانه بررسی شده است. در مدل انتشارات، موجودیت‌هایی نظیر Kind، Publication و PublicationTool تعریف شده‌اند که بیانگر اطلاعات انتشارات و ابزارهای انتشار هستند. مدل مقصد نیز شامل LibraryRoot و Recording است که ساختار ساده‌ای برای ذخیره رکوردهای کتابخانه‌ای ارائه می‌دهد. تبدیل

مدل از طریق سه قانون انجام شده است؛ به‌گونه‌ای که هر PublicationTool به LibraryRoot و هر Publication به Recording تبدیل می‌شود و اطلاعات نوع انتشارات نیز منتقل می‌گردد. این مطالعه نشان می‌دهد که با استفاده از این قوانین می‌توان به سادگی اطلاعات انتشارات را به ساختار کتابخانه‌ای نگاشت و مدیریت نمود.

۴-۱-۲- تبدیل درخت نحو انتزاعی به گراف جهت‌دار بدون دور

در این مطالعه موردی [۱۸]، فرایند تبدیل مدل درخت نحو انتزاعی (AST) به گراف جهت‌دار بدون دور (DAG) بررسی شده است. مدل مبدا مبتنی بر فرامدل ExpressionAST طراحی شده است که موجودیت‌هایی نظیر Node و Expression را شامل می‌شود. هر گره می‌تواند به چندین گره دیگر متصل شود و ساختار یک درخت را شکل دهد.

مدل مقصد بر اساس فرامدل ExpressionDAG ساخته شده که با بهینه‌سازی ساختار درختی به شکل یک گراف جهت‌دار بدون دور (DAG)، تکرار و افزونگی در گره‌ها را کاهش می‌دهد و ارتباطات را فشرده‌تر می‌کند.

در فرایند تبدیل، گره‌های مشترک در AST شناسایی شده و در DAG به صورت یکتا مدل می‌شوند. همچنین مرزهای بیانگر ارتباط گره‌ها در DAG به‌گونه‌ای تنظیم می‌شوند که مسیرهای موجود در AST حفظ گردد. تبدیل مدل با هدف بهبود بهره‌وری در پردازش و ذخیره‌سازی ساختارهای نحوی پیچیده انجام شده و نشان‌دهندهٔ قدرت روش یادگیری تقویتی در نگاشت دقیق ساختارهای داده‌ای متفاوت است.

این مطالعه موردی نشان می‌دهد که الگوریتم کیوموتور با بهره‌گیری از یادگیری تقویتی، بدون نیاز به تعریف دستی قوانین تبدیل پیچیده، قادر به استخراج خودکار و بهینه نگاشت‌های چند به یک بین درخت نحو انتزاعی (AST) و گراف جهت‌دار بدون دور (DAG) است. این الگوریتم با ساخت و به‌روزرسانی جدول Q در طول اپیزودها، ساختارهای پیچیده AST را به‌طور دقیق به DAG بهینه‌شده تبدیل می‌کند، در حالی که افزونگی‌ها را حذف و قوانین ترتیبی مسیرها را حفظ می‌کند.

۴-۱-۳- تبدیل یک نمودار گانت به یک شبکهٔ مسیر بحرانی

در این مطالعه موردی [۱۸]، فرایند تبدیل یک نمودار گانت (Gantt) به یک شبکهٔ مسیر بحرانی (CPM) به‌عنوان یک سناریوی گرافی وابستگی‌محور بررسی شده است.

مدل مبدأ مبتنی بر فرامدل GanttModel شامل موجودیت‌هایی نظیر Task و Dependency است که توالی و زمان‌بندی وظایف پروژه را نمایش می‌دهند. هر Task دارای بازهٔ زمانی مشخص بوده و می‌تواند به یک یا چند وظیفه دیگر وابسته باشد، به‌گونه‌ای که ساختار منطقی پروژه در قالب یک گراف وظیفه‌گرا شکل می‌گیرد.

مدل مقصد نیز مطابق با فرامدل CPMModel طراحی شده و دارای کلاس‌هایی مانند CPMActivity و CPMRelation می‌باشد که مسیرهای بحرانی و وابستگی‌های زمانی را به‌صورت گراف جهت‌دار مدل‌سازی می‌کنند. در این تبدیل، برای هر وظیفهٔ Task در مدل مبدأ، یک CPMActivity متناظر در مدل مقصد ساخته می‌شود و هر رابطهٔ Dependency نیز به یک CPMRelation متناظر تبدیل می‌گردد. بنابراین، نگاشت بین عناصر عمدتاً به‌صورت یک‌به‌یک انجام می‌شود.

با این حال، چالش اصلی این تبدیل نه در نگاشت اسمی عناصر، بلکه در حفظ ساختار وابستگی و مسیرهای زمانی میان فعالیت‌هاست. الگوریتم کیوموتور با استفاده از یادگیری تقویتی، نگاشت‌هایی را ترجیح می‌دهد که ساختار مسیرهای پیش‌نیاز و پس‌نیاز را حفظ کرده و به صورت خودکار از طریق تابع پاداش، تطابق دقیق این روابط را فرا می‌گیرد. این مطالعه موردی نشان‌دهنده‌ی توانایی الگوریتم در مدیریت ساختارهای گرافی پیچیده بدون تعریف قوانین دستی نگاشت می‌باشد.

۴-۱-۴- تبدیل شبکه پتری وزن‌دار به شبکه پتری بدون وزن

در ایـــن مطالعه موردی [۱۸]، فرایند تبدیل یک شـــبکهٔ پتری وزن‌دار (Weighted Petri Net) به یک شـــبکهٔ پتری بدون وزن (Unweighted Petri Net) بررسی شده است.

مدل مبدأ، مبتنی بر فرامدل WeightedPetriNet، شامل موجودیت‌هایی نظیر Place، Transition و Edge می‌باشـــد که در آن‌ها علاوه بر ساختار اتصال گره‌ها، ویژگی عددی weight نیز برای هر یال تعریف شـــده اســـت. این ویژگی بـــرای تعیین میزان تأثیر یـــک انتقال در جریـــان توکن‌ها استفاده می‌شود.

در مقابـــل، مدل مقصد مطابق با فرامـــدل Unweighted-edPetriNet طراحی شده و شـــامل همان موجودیت‌های ساختاری اســـت، با این تفاوت که در کلاس‌های رابطه‌ای، ویژگی weight تعریف نشـــده اســـت و مدل صرفاً جریان منطـــقی بین گره‌هـــا را بدون در نظر گرفتـــن وزن نمایش می‌دهد.

در این تبدیل، از آنجا که ویژگی weight در مدل مقصد وجود ندارد، این ویژگی به‌صورت خودکار از نگاشـــت‌ها حذف می‌شـــود. الگوریتم کیوموتور با تشـــخیص تفاوت ســـاختار فرامدل‌ها و بدون نیاز بـــه تعریف قوانین حذف صریح، این ویـــژگی را در فرآیند یادگیری نادیده گرفته و نگاشـــت عناصر را صرفاً بر اســـاس ویژگی‌های مشترک انجام می‌دهد.

این مطالعه موردی نشـــان می‌دهد که کیوموتور قادر اســـت تبدیل‌هایی را که شامل حذف ویژگی‌های ساختاری هســـتند، به‌صورت ســـازگار با مدل مقصد و بدون تولید نگاشـــت ناســـازگار انجام دهد. ایـــن توانـــایی به‌ویژه در سناریوهایی که مدل مقصد ســـاختار ساده‌تری نسبت به مدل مبدا دارد، اهمیت دارد.

۴-۱-۵- تبدیل مجموعه به مجموعه مرتب

در ایـــن مطالعه مـــوردی [۱۸]، فراینـــد تبدیل مدل‌های مجموعه‌ای (Set) به مدل‌هـــای مجموعه مرتب (Ordered Set) بررسی شـــده اســـت. مدل مبدا بر پایهٔ فرامدل Sets طـــراحی شـــده کـــه در آن موجودیت‌هـــایی نظیـــر Set و Element تعریف شـــده‌اند. در ایـــن مدل، عناصر مجموعه بدون هیچ ترتیب مشـــخصی نگهداری می‌شـــوند و صرفاً عضویت آن‌ها اهمیت دارد.

مدل مقصد بر اســـاس فرامدل OrderedSets ســـاخته شده است که علاوه بر نگهداری عناصر، ترتیب قرارگیری آن‌ها را نیز ذخیره می‌کند. هر Element در مجموعه مرتب دارای ویژگی index اســـت که موقعیـــت عنصر در ترتیب مجموعه را تعیین می‌کند.

در فراینـــد تبدیل، عناصر موجـــود در مجموعهٔ اصلی اســـتخراج شـــده و به‌گونه‌ای در مجموعهٔ مرتب نگاشـــت می‌یابند که ترتیب مشـــخصی برای آن‌ها ایجاد شـــود. در صورتی که ترتیب خاصی از پیش تعریف نشـــده باشـــد، ترتیب پیش‌فرض بر اســـاس ترتیـــب ورود عناصر تعیین می‌شود.

این تبدیـــل با هدف افـــزودن قابلیت‌هـــای جدید برای عملیات مبتنی بر ترتیب بر روی داده‌ها انجام شـــده است. نتایج ارزیابی نشـــان داد که الگوریتم کیوموتور توانســـته اســـت بدون نیاز به تعریف قوانیـــن صریح، مجموعه‌های نامرتب را به‌درستی به نسخه‌های مرتب آن‌ها تبدیل کند.

۴-۱-۶- تبدیل UML به Java

در این مطالعه مـــوردی [۱۶]، فرآینـــد تبدیل مدل‌های UML به مدل‌های Java بررسی شـــده اســـت. مدل مبدأ بر پایهٔ فرامدل ساده‌شـــدهٔ UML طراحی شده است که شامل موجودیت‌هـــایی ماننـــد Class، Package و Attribute می‌باشد. هر Package شامل مجموعه‌ای از کلاس‌ها بوده و هر Class می‌تواند ویژگی‌هایی مانند name و ID را دربر گیرد.

مدل مقصد نیز بر اساس فرامدل ساده‌شدهٔ Java طراحی شده است که ساختاری مشابه دارد، اما از موجودیت‌هایی مانند JavaClass، JavaPackage و Field اســـتفاده می‌کند.

در این تبدیل، ویژگی‌های کلاس‌های UML به حوزه‌های کلاس‌های Java نگاشت داده شده‌اند و سلسله‌مراتب بسته‌ها نیز به‌درستی حفظ شده است.

با این حال، لازم به ذکر است که برخی قراردادهای رایج در زبان Java نظیر روش‌های سازنده (constructor)، روش‌های دستیابی (getter/setter)، سطح دسترسی (access modifiers) یا قواعد نگاشت نوع داده‌های پیچیده، به‌صورت صریح در فرامدل UML تعریف نشده‌اند و در نسخه ساده‌شدهٔ این مطالعه موردی نیز لحاظ نگردیده‌اند.

از این‌رو، الگوریتم کیوموتور در این تبدیل صرفاً بر اساس اطلاعات صریح موجود در فرامدل‌ها عمل می‌کند و نگاشت‌هایی را یاد می‌گیرد که بین عناصر مدل‌سازی‌شده به‌طور مستقیم قابل تطبیق باشند.

این مطالعه موردی با هدف تسهیل فرایند تولید خودکار کد از مدل‌های مفهومی طراحی شده و نشان می‌دهد که روش پیشنهادی توانسته بدون نیاز به تعریف قوانین نگاشت دستی، ساختار کلی مدل را به‌درستی یاد بگیرد و اعمال کند، هر چند پوشش قراردادهای ضمنی زبان مقصد نیازمند توسعه‌ی فرامدل یا تعریف قواعد تولیدی مستقل خواهد بود.

۴-۱-۷- تبدیل نمودار کلاس به شمای رابطه‌ای

در این مطالعه موردی [۱۹]، تبدیل مدل مفهومی نمودار کلاس (Class Diagram) به مدل داده‌ای شمای رابطه‌ای (Relational Schema) بررسی شده است.

مدل مبدأ بر اساس فرامدل ساده‌شدهٔ ClassDia-gram طراحی شده است که شامل عناصر اصلی مانند Attribute، Class و DataType می‌باشد. هر Class می‌تواند چندین Attribute داشته باشد که هرکدام دارای نوع دادهٔ مشخص هستند و در مواردی ساختارهای ترکیبی نیز بین کلاس‌ها برقرار است.

مدل مقصد بر مبنای فرامدل RelationalSchema ساخته شده است که ساختارهای پایگاه‌داده‌ای مانند Column، Table، Schema و Type را شامل می‌شود. در این مدل، کلاس‌ها به جدول‌ها، ویژگی‌ها به ستون‌ها و نوع داده‌ها به انواع داده‌های رابطه‌ای متناظر نگاشت می‌یابند. همچنین ساختار سلسله‌مراتبی مدل مبدأ در قالب شیء Schema بازنمایی می‌شود.

در فرآیند تبدیل، علاوه بر نگاشت‌های پایه، نگاشت‌هایی مانند تبدیل نوع دادهٔ ویژگی‌ها، ایجاد کلید اولیه برای جدول‌ها، و حفظ روابط ترکیبی بین کلاس‌ها نیز به‌طور خودکار انجام شده است.

الگوریتم Q-Motor با دریافت تنها یک نمونهٔ آموزشی از مدل مبدأ و مقصد، توانست الگوهای ساختاری را از طریق فرآیند یادگیری تقویتی استخراج کرده و نگاشت‌هایی فراتر از تطبیق مستقیم کلاس به جدول و ویژگی به ستون را شناسایی و تعمیم دهد.

۴-۱-۸- تبدیل نسخه ای از پایگاه دادهٔ افراد به نسخهٔ جدیدتر

در این مطالعه موردی [۱۸]، فرآیند مهاجرت داده از نسخهٔ اولیهٔ پایگاه داده اشخاص (PersonsDB1) به نسخهٔ جدیدتر آن (PersonsDB2) بررسی شده است. مدل مبدا شامل موجودیت‌هایی مانند Person با ویژگی‌هایی مانند name و age بوده و ساختاری ساده برای نگهداری اطلاعات اشخاص ارائه می‌کرد.

مدل مقصد در نسخه جدید، علاوه بر نگهداری اطلاعات پایه، ویژگی‌هایی مانند email، gender یا ساختار سلسله‌مراتبی جدیدی برای دسته‌بندی افراد را نیز دربرمی‌گیرد. این توسعه در فرامدل باعث افزایش دقت و قابلیت تحلیل داده‌ها شده است.

در فرآیند تبدیل، ویژگی‌های مشترک مانند name و age به‌صورت مستقیم منتقل شده‌اند و برای ویژگی‌های جدید (مانند gender یا email)، مقداردهی پیش‌فرض یا استنتاجی انجام گرفته است.

این نوع تبدیل در دستهٔ مهاجرت مدل‌های پایگاه داده‌ای تک‌دامنه‌ای قرار می‌گیرد و نشان می‌دهد که الگوریتم

پیشنهادی می‌تواند بدون نیاز به نوشتن قوانین دستی، بین نسخه‌های مختلف یک مدل داده مهاجرت انجام داده و سازگاری ساختاری و معنایی را حفظ کند.

۴-۱-۹- تبدیل کتاب و مقاله به انتشارات

در این مطالعه موردی، تبدیل ساختار مدل ترکیبی کتاب و مقاله به مدل انتشارات مورد بررسی قرار گرفته است. مدل مبدا شامل موجودیت‌هایی مانند Book، Article، Author و Chapter است که ساختار پیچیده‌ای با روابط چندبه‌چند را ایجاد می‌کند.

مدل مقصد شامل موجودیت‌های Publication و Publishers است که ساختاری ساده‌تر و یکپارچه‌تر دارد و اطلاعات کتاب‌ها و مقالات را در قالب انتشارات ذخیره می‌کند.

در فرآیند تبدیل، کتاب‌ها، مقالات، فصل‌ها و نویسندگان به انتشارات و ویژگی‌های متناظرشان نگاشت شدند. به دلیل وجود نگاشت‌های چندبه‌چند، این مطالعه موردی چالشی‌تر از موارد قبلی بود و نشان داد که الگوریتم پیشنهادی در مدیریت نگاشت‌های پیچیده نیاز به بهبود دارد. با این حال، نگاشت بخش عمده‌ای از ساختار مدل به درستی انجام شده است و قابلیت الگوریتم برای تطبیق ساختارهای پیچیده به طور تجربی مورد ارزیابی قرار گرفت.

۴-۱-۱۰- تبدیل یک کیسه به نسخهٔ بهبود یافته‌ای از آن

در این مطالعه موردی [۱۸]، تبدیل از یک نمایه کیسه‌ها (Bags) مبتنی بر فرامدل Bags1 به نمایه‌ای بهبودیافته مبتنی بر فرامدل Bags2 بررسی شده است. در فرامدل Bags1، هر تکرار یک مقدار توسط یک شیء جداگانه با ویژگی value نمایش داده می‌شود. در فرا مدل Bags2، برای هر مقدار یکتای value تنها یک شیء وجود دارد که ویژگی multiplicity، تعداد تکرارهای آن مقدار را مشخص می‌کند. هدف این تبدیل، نگاشت اشیاء و ویژگی‌های آن‌ها از Bags1 به Bags2 است، به‌طوری که اطلاعات مدل مبدا حفظ شده و در قالب جدید با ویژگی‌های اضافی بازنمایی شود.

فرآیند تبدیل شامل گروه‌بندی اشیاء با مقدار یکسان در Bags1 و ایجاد یک شیء در Bags2 با ویژگی multiplicity است که تعداد این اشیاء را نشان می‌دهد. ویژگی multiplicity، که در Bags1 وجود ندارد، چالشی برای نگاشت خودکار توسط الگوریتم کیوموتور ایجاد می‌کند. کیوموتور با استفاده از یادگیری تقویتی، تلاش می‌کند این ویژگی جدید را از طریق جفت‌های آموزشی نمونه کشف و نگاشت کند.

این مطالعه موردی توانایی کیوموتور را در مدیریت تبدیل‌هایی با ویژگی‌های جدید آزمایش می‌کند، اگرچه چالش‌های مرتبط با استخراج multiplicity بر عملکرد آن تأثیر گذاشته است، که در بخش‌های بعدی تحلیل می‌شود.

۴-۲- نتایج ارزیابی معیارها

جدول ٤ نتایج ارزیابی عملکرد الگوریتم کیوموتور را برای ده مطالعه موردی مختلف نشان می‌دهد که معیارهای دقت، حساسیت، صحت، امتیاز F1 و ضریب همبستگی متیوز (MCC) را گزارش می‌کند. در هشت مطالعه‌موردی اول، از جمله «تبدیل انتشارات به کتابخانه» تا «تبدیل نسخه‌ای از پایگاه دادهٔ افراد به نسخهٔ جدیدتر»، همه معیارها مقدار ۱۰۰ درصد را نشان می‌دهند که به MCC برابر با ۱ منجر شده است. این نتایج حاکی از عملکرد کامل الگوریتم در این موارد است، به‌طوری که هیچ خطای مثبت کاذب یا منفی کاذب وجود ندارد و الگوریتم توانسته است نگاشت کامل و معتبری بین گراف مدل میانی و گراف مدل مقصد ایجاد کند. این نتایج بیانگر توانایی بالای الگوریتم در مدیریت نگاشت‌های یک‌به‌یک و ساختارهای با پیچیدگی متوسط بدون تولید خطا (مثبت یا منفی کاذب) است.

با این حال، در دو مطالعه موردی آخر، یعنی «تبدیل کتاب و مقاله به انتشارات» و «تبدیل یک کیسه به نسخهٔ

بهبـــود یافتـــه‌ای از آن»، عملکرد الگوریتـــم کاهش یافته و معیارهـــا مقادیر کمتری را نشـــان می‌دهند. به‌طور خاص، در مـــورد «تبدیل کتاب و مقاله به انتشـــارات»، با دقت ٦٩ درصد، حســـاسیت ٨٠ درصد، صحت ٧٤ درصد و امتیاز F1 برابـــر با ٧٤ درصد، مقـــدار MCC بـــه ٠/٤٨٨ رسیده است، که نشـــان‌دهنده وجود چالش‌هایی در ایجاد نگاشت کامل است. به‌طور مشـــابه، در «تبدیل یک کیسه به نسخهٔ بهبود یافتـــه‌ای از آن»، با دقت ٨٨ درصد، حســـاسیت ٧٢ درصـــد، صحت ٧٩ درصد و امتیاز F1 برابر با ٧٩ درصد، MCC به ٠/٥٩٧ رسیده اســـت. ایـــن مقادیر پایین‌تر به‌دلیل چالش‌های نگاشت‌های چندبه‌چند و تغییرات ساختاری در فرامدل مقصد است. این افت عملکرد نشان‌دهنده محدودیت الگوریتم در شناسایی الگوهای پیچیده و نیاز به بهبود در سناریوهای با روابط متنوع است.

جدول (۴): مقادیر به‌دست آمده برای معیارها

مطالعه موردی‌ها	دقت	حساسیت	صحت	F1	MCC
تبدیل انتشارات به کتابخانه	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱
تبدیل درخت نحو انتزاعی به گراف جهت‌دار بدون دور	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱
تبدیل تبدیل یک نمودار گانت به یک شبکه‌ی مسیر بحرانی	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱
تبدیل شبکه پتری وزن‌دار به شبکه پتری بدون وزن	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱
تبدیل مجموعه به مجموعه مرتب	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱
تبدیل UML به Java	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱
تبدیل نمودار کلاس به شمای رابطه‌ای	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱
تبدیل نسخه‌ای از پایگاه داده‌ی افراد به نسخه‌ی جدیدتر	۱۰۰	۱۰۰	۱۰۰	۱۰۰	۱
تبدیل کتاب و مقاله به انتشارات	۶۹	۸۰	۷۴	۷۴	۰/۴۸۸
تبدیل یک کیسه به نسخه‌ی بهبود یافته‌ای از آن	۸۸	۷۲	۷۹	۷۹	۰/۵۹۷

۴-۳- مقایسهٔ مقیاس‌پذیری روش پیشنهادی با زبان ETL و ATL

تبدیل مورد استفاده برای ارزیابی عملکرد در شکل ۸، «تبدیل UML به جاوا» اســـت کـــه در بخش ٤-١-٦ معرفی شـــده است. این تبدیل به دلیل ســـادگی ساختار فرامدل‌ها و نگاشـــت‌های یک‌به‌یک انتخاب شـــد تا تمرکز ارزیابی بر مقیاس‌پذیـــری الگوریتم‌های کیوموتور، زبان اســـتاندارد تبدیل اپسیلون (ETL) [۱۱] و ATL [۹] باشد، نه بر پیچیدگی ســـاختاری یا قدرت بیان تبدیل. در این مطالعه موردی، که در آن هـــر عنصر از مدل مبدا به یک عنصر از مدل مقصد نگاشته می‌شود و پیچیدگی‌های ساختاری در فرامدل‌ها را ندارد، امکان مقایسه عادلانه عملکرد زمانی را در مدل‌های با اندازه‌های مختلف فراهـــم کرده و امکان بررسی تطبیق مدل تولید شده با مدل مقصد را فراهم می‌کند. مدل‌های مبدا (UML) و مقصد (جاوا) با استفاده از اسکریپت‌های پایتون تولید شدند و با طراحی دقیق، تناظر کامل بین مقادیر مبدا و مقصد تضمین شد. در نتیجه، هر سه روش (کیوموتور، ETL و ATL) مدل‌های مقصد را با صحت کامل تولید کردند و انطباق مدل‌های تولید شده با مدل‌های مقصد بررسی و تأیید شد.

برای اجرای این آزمون، الگوریتم کیوموتور ابتدا روی یـــک جفت نمونـــه کوچک از تبدیل UML به جـــاوا با زمان پیش‌آموزش ٥ میلی‌ثانیه آموزش دیده و قوانین تبدیل را در جدول Q ذخیره کرده است. این مرحله پیش‌آموزش به‌دلیل زمان اجـــرای بسیار کوتاه، تأثیـــر قابل‌توجهی بر عملکرد کلی ندارد. در مرحلـــه آزمون، کیوموتور از قوانین ذخیره شـــده در جدول Q برای نگاشت مدل‌های بزرگ‌تر استفاده می‌کند، که این رویکرد به آن امکان حفظ مقیاس‌پذیری بالا با افزایش اندازه مدل را می‌دهد.

در ایـــن نمودار، محور افقی نشـــان‌دهندهٔ «اندازهٔ مدل» ورودی (در مقیاس لگاریتمی) اســـت که به عنوان معیاری از پیچیدگی مسئله در نظر گرفته می‌شود و محور عمودی، «زمان اجرا» هر الگوریتم را به ثانیه و در مقیاس لگاریتمی نشـــان می‌دهد. منحنی آبی مربوط به عملکرد کیوموتور و منحنی قرمز مربوط به عملکرد ETL و منحنی سبز مربوط به عملکرد ATL می‌باشـــد. همانطور که مشـــاهده می‌شود، با افزایش اندازهٔ مدل، زمان اجرای هر ســـه روش به‌طور طبیـــعی افزایش می‌یابـــد. برای مدل‌های بـــا اندازهٔ کوچک (کمتـــر از ۱۰/۰۰۰ واحد)، زمان اجرای ســـه روش تفاوت

قابـــل ملاحظه‌ای ندارد و حتی در بـــرخی نقاط، ETL اندکی سریع‌تر عمل می‌کند. اما با افزایش اندازهٔ مدل، روند تغییر می‌کنـــد و کارایی الگوریتم کیوموتور به‌طور چشـــمگیری بهتـــر از ETL و ATL ظاهر می‌شـــود. به‌طور مشـــخص، از انـــدازهٔ مدل ۱۰/۰۰۰ به بعد، شیب افزایش زمان اجرا برای کیوموتور به مراتب کمتر از ETL است. این امر نشان‌دهندهٔ مقیاس‌پذیری بهتر روش کیوموتور در مواجهه با مســـائل بزرگ‌تر اســـت. برای مثال، در اندازهٔ مدل ۵۰/۰۰۰، زمان اجرای کیوموتور حدوداً ۱ ثانیه اســـت، در حالی که ETL و ATL بـــه زمانی نزدیک به ۵ ثانیه نیاز دارد. این اختلاف در اندازهٔ مدل ۲۵۰/۰۰۰ بسیار بیشـــتر شـــده و زمان اجرای کیوموتـــور حدود ۸ ثانیه و زمان اجـــرای ETL بیش از ۳۰ ثانیه و زمان اجرای ATL حدود ۱۵ ثانیه است.

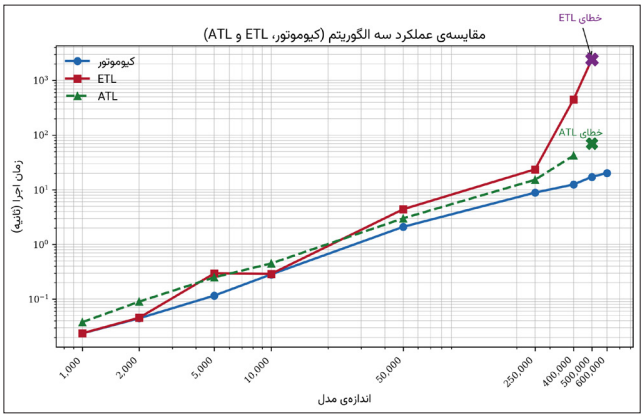
نکتهٔ حائز اهمیت دیگر، عملکرد الگوریتم‌ها در بزرگترین اندازهٔ مدل آزمون شـــده (۶۰۰/۰۰۰) اســـت. در این نقطه، الگوریتم کیوموتور با موفقیت اجرا شـــده و زمانی حدود ۲۰ ثانیه را ثبت کرده اســـت، امـــا روش ETL در این اندازه با خطا مواجه شـــده و قادر به تکمیل پردازش نبوده است (نقطهٔ مشـــخص شـــده با عنوان «خطای ETL»). روش ATL در انـــدازهٔ مدل ۵۰۰/۰۰۰ دچار خطا شـــده اســـت و دیگر نمی‌تواند پردازش را کامل کند (نقطه‌ی مشـــخص شـــده با عنوان «خطای ATL»).

در مجموع، نتایج این نمودار به وضوح برتری الگوریتم کیوموتور را از نظر ســـرعت اجـــرا و مقیاس‌پذیری برای مدل‌های با اندازهٔ متوسط و بزرگ نشان می‌دهد. علاوه بر این، پایداری کیوموتور در پردازش مدل‌های بسیار بزرگ، در حالی که ETL و ATL با شکســـت مواجه می‌شوند، مزیت کاربردی مهمی برای روش پیشـــنهادی مبتنی بر یادگیری تقویتی محسوب می‌گردد.

۵- بحث

۵-۱- قوانین توصیفی تبدیل

استفاده از قوانین توصیفی و دستی برای تبدیل مدل‌ها



شکل (۸): زمان اجرای ETL و ATL در مقایسه با کیوموتور

بسیـــار وقت‌گیر اســـت. این فرایند نیازمنـــد درک عمیق از ســـاختار و معنای مدل‌های مبدا و مقصد و همچنین دانش کافی از زبان‌های تبدیل مدل برای تدوین این قوانین است. علاوه بر ایـــن، تلاش زیادی برای نگهداری و توصیف این تبدیل‌هـــا در محیط دیجیتالی که به ســـرعت تغییر می‌کند، مورد نیاز است.

۵-۲- همترازی دستی بین مدل‌های مبدا و مقصد

بسیاری از پژوهش‌ها بر اســـاس همترازی دســـتی از پیش تعیین‌شـــده بین عناصر مدل‌های مبدا و مقصد انجام می‌شـــود. این همترازی همیشـــه آســـان نیســـت و گاهی نیازمند تخصص و تجربه زیادی است. تأکید شده که نبود خودکارسازی، یکی از شکاف‌های بزرگ در این حوزه است و باعث محدود شدن قابلیت تعامل بین مدل‌ها می‌شود.

۵-۳- کاربرد شبکه‌های عصبی در یادگیری تبدیل مدل‌ها

تحقیقات نشان داده که یادگیری تبدیل مدل‌ها با استفاده از شـــبکه‌های عصبی مصنوعی نیازمند حجـــم زیادی از داده‌های آموزشی اســـت [۱۵]. این روش به دلیل پیچیدگی روابـــط بین مدل‌های مبدا و مقصـــد، محدودیت‌هایی دارد. علاوه بر این، تنوع داده‌های آموزشی و تعداد مدل‌های در مجموعـــه داده‌ها نیز نقش مهمی در موفقیت این فرایند ایفا می‌کنند.

۶- نتیجه‌گیری

در این پژوهش، یک روش نوین مبتنی بر یادگیری تقویتی برای تبدیل خودکار مدل‌ها ارائه شد. نتایج ارزیابی نشان داد که رویکرد پیشنهادی، علاوه بر کاهش نیاز به مداخلهٔ دستی، دقت بالاتری در نگاشت عناصر مدل‌ها دارد و از نظر زمان اجرا و مقیاس‌پذیری عملکرد بهتری نسبت به روش‌های سنتی مانند زبان ETL و ATL ارائه می‌دهد. به‌ویژه، این روش توانایی انطباق پویا با تغییرات مدل‌های مبدا و مقصد را دارد و می‌تواند بدون نیاز به داده‌های آموزشی گسترده، بهینه‌ترین قوانین تبدیل را استخراج کند

با وجود این مزایا، روش پیشنهادی همچنان با چالش‌هایی مواجه است، از جمله نیاز به بهبود دقت در سناریوهای پیچیده‌تر و امکان افزایش کارایی از طریق ترکیب با روش‌های یادگیری عمیق. در پژوهش‌های آینده، می‌توان با بهبود قابلیت یادگیری الگوهای پیچیده‌تر و طراحی سیستم‌های خودکار برای ارزیابی کیفیت تبدیل مدل‌ها، کارایی این روش را افزایش داد. همچنین، گسترش دامنهٔ کاربرد این رویکرد در حوزه‌های دیگر مانند مهندسی سیستم‌ها و پردازش داده‌های بزرگ می‌تواند مسیر جدیدی برای تحقیق و توسعه در این زمینه فراهم کند.

یکی از چالش‌های کلیدی در یادگیری تقویتی، تعریف دقیق تابع پاداش است که بر دقت و سرعت همگرایی عامل یادگیری تأثیر می‌گذارد. برای بهبود این بخش، پیشنهاد می‌شود از مدل‌های زبانی بزرگ (LLMs) برای ایجاد یک تابع پاداش پویا و هوشمند استفاده شود. حتی از LLMها می‌توان برای یافتن بهترین جفت مدل‌های نمونهٔ اولیه نیز بهره گرفت. مدل‌های زبانی بزرگ می‌توانند با تحلیل جفت مدل‌های آموزشی، الگوهای نگاشت را شناسایی کرده و پاداش‌هایی متناسب با صحت نگاشت، بهینه‌سازی ساختاری (مانند ادغام گره‌های مشترک) و حفظ مسیرهای بحرانی تولید کنند. این فرایند شامل استخراج ویژگی‌های مدل، ارزیابی کیفیت نگاشت‌ها و تنظیم پویای پاداش‌ها بر اساس داده‌های ورودی است. از نظر عملی، این رویکرد با استفاده از زیرساخت‌های محاسباتی موجود (مانند کارسازهای GPU یا APIهای مدل‌های زبانی بزرگ) امکان‌پذیر است، اگرچه نیاز به منابع محاسباتی قابل‌توجهی دارد. برای کاهش این چالش، می‌توان از مدل‌های سبک‌تر یا روش‌های انتقال یادگیری استفاده کرد. یکپارچه‌سازی مدل‌های زبانی بزرگ می‌تواند دقت و تعمیم‌پذیری روش پیشنهادی را، به‌ویژه در سناریوهای پیچیده، بهبود بخشد و با پردازش موازی، مقیاس‌پذیری الگوریتم را افزایش دهد. در کارهای آینده، بررسی تنظیم دقیق مدل‌های زبانی بزرگ برای تبدیل‌های افزایشی و ارزیابی خودکار کیفیت نگاشت‌ها پیشنهاد می‌شود.

مراجع

[1]Sharbaf, M., Zamani, B. & Sunyé, G. 2022."Automatic resolution of model merging conflicts using quality-based reinforcement learning," Journal of Computer Languages, Vol. 71, p. 101123.

[2]Batot E. R. & Sahraoui, H. 2022. "Promoting social diversity for the automated learning of complex MDE artifacts," Software and Systems Modeling, Vol. 21, No. 3, pp. 1159-1178, doi: 10.1007/s10270-021-00969-9.

[3]Akdur, D., Garousi, V. & Demirörs, O. 2018. "A survey on modeling and model-driven engineering practices in the embedded software industry," Journal of Systems Architecture, Vol. 91, pp. 62-82.

[4]Höppner, S., Kehrer, T. & Tichy, M. 2021. "Contrasting dedicated model transformation languages versus general purpose languages: a historical perspective on ATL versus Java based on complexity and size," Software and Systems Modeling, Vol. 21, No. 2, pp. 805-837, doi: 10.1007/s10270-021-00937-3.

[5]Rouhi, A., Kolahdouz Rahimi, S. & Lano, K. 2022. "Formalizing model transformation patterns," Journal of Software: Evolution and Process, Vol. 34, No. 2, p. e2406.

[6]Jilani, A. A., Khan, M. U., Iqbal, M. Z. & Usman, M. 2022. "An automated search-based test model generation approach for structural testing of model transformations," Journal of Software: Evolution and Process, Vol. 34, No. 11, p. e2461, 2022.

[7]Weidmann, N., Yigitbas, E., Anjorin, A., Srivastava, A. & Jose, J. 2022. "Human-in-the-Loop Large-Scale Model Transformations with the VICToRy Debugger," J. Object Technol., Vol. 21, No. 3, pp. 3:1-15.

[8]Höppner, S., Haas, Y., Tichy, M. & Juhnke, K. 2022. "Advantages and disadvantages of (dedicated) model transformation languages: A qualitative interview study," Empirical Software Engineering, Vol. 27, No. 6, p. 159.

[9]Jouault, F., Allilaire, F., Bézivin, J. & Kurtev, I. 2008. “ATL: A model transformation tool,” Science of computer programming, Vol. 72, No. 1-2, pp. 31-39.
[10]Eisenberg, M., Pichler, H.-P., Garmendia, A. & Wimmer, M. 2021. “Towards reinforcement learning for in-place model transformations," in 2021 ACM/IEEE 24th International Conference on Model Driven Engineering Languages and Systems (MODELS), IEEE, pp. 82-88.
[11]Kolovos, D. S., Paige, R. F. & Polack, F. A. 2008.”The epsilon transformation language,” in Theory and Practice of Model Transformations: First International Conference, ICMT 2008, Zürich, Switzerland, July 1-2, 2008 Proceedings 1, 2008: Springer, pp. 46-60.
[12]Lano K. & Xue, Q. 2022. “Code Generation by Example,” in MODELSWARD, pp. 84-92.
[13]Lano, K., Kolahdouz-Rahimi, S. & Fang, S. 2021. “Model transformation development using automated requirements analysis, metamodel matching, and transformation by example,” ACM Transactions on Software Engineering and Methodology (TOSEM), Vol. 31, No. 2, pp. 1-71.
[14]Eisenberg, M.& Wimmer, M. 2024. “From single-objective to multi-objective reinforcement learning-based model transformation,” Software and Systems Modeling, pp. 1-31.
[15]Burgueno, L., Cabot, J., Li, S. & Gérard, S. 2022. “A generic LSTM neural network architecture to infer heterogeneous model transformations,” Software and Systems Modeling, Vol. 21, No. 1, pp. 139-15.
[16]Nikbakht-Nasrabadi, D., Samimi-Dehkordi, L. & Basiri, M. 2025. “A Preliminary Investigation into Automated Model Transformation Using Q-Learning: A Case Study,” presented at the 29th International Conference of the Computer Society of Iran (CSICC 2025), Sharif University of Technology, Tehran.
[17]Samimi-Dehkordi, L., Zamani, B. & Kolahdouz-Rahimi, S. 2018.”EVL+ Strace: a novel bidirectional model transformation approach," Information and Software Technology, Vol. 100, pp. 47-72.
[18]Westfechtel, B. 2018. “Case-based exploration of bidirectional transformations in QVT relations,” Software & Systems Modeling, Vol. 17, pp. 989-1029.
[19]Willink, E. D. 2003.”A concrete UML-based graphical transformation syntax: The UML to RDBMS example in UMLX,” Metamodelling for MDA, p. 129.