arXiv:2301.00623v1 [cs.SE] 2 Jan 2023

面向多版本模型的三元图文法

MatthiasBarkowsky Holger Giese
matthias.barkowsky@hpi.de holger.giese@hpi.de

2023年1月3日

摘要

与传统软件项目一样，模型驱动软件工程中的项目也需要对多个版本的开发制品进行妥善管理，尤其要能够应对暂时性不一致。在模型驱动软件工程中，所采用的版本管理方法还必须处理这样的情况：不同的制品，即不同的模型，通过自动模型转换相互关联。

在本报告中，我们提出了一种联合处理将源模型的多个版本转换为目标模型相应版本的技术。该技术支持采用更紧凑的表示形式，从而有望改善转换本身及后续分析操作的执行时间。我们的方法基于广为人知的三元图文法形式化体系，以及一种先前提出的、称为多版本模型的模型版本历史编码方法。除了证明我们的方法相对于三元图文法标准语义的正确性外，我们还开展了实证评估，结果显示该方法在执行时间性能方面具有潜在优势。

1 引言

在模型驱动软件开发中，模型被视为主要的开发制品。复杂项目可能涉及多个模型，这些模型从不同的抽象层次描述正在开发的系统，或针对不同的系统方面进行描述，并且团队中的开发人员可以独立编辑这些模型。在这种情况下，通过模型转换来确保整体系统描述的一致性：模型转换会自动从更抽象的模型推导出具体模型，或将描述系统某一方面的模型中的变更传播到涉及不同但相互重叠方面的其他模型中 [20]。

与传统软件开发中的程序代码类似，不同开发者所做的更改会推动模型演进，因此需要管理由此产生的软件描述版本。具体而言，版本管理必须支持多个开发者围绕同一开发制品开展并行开发活动；为了避免信息丢失，有时可能需要暂时容忍单个制品中的不一致{v1}。

在[2], 中，我们引入了多版本模型，用于管理同一模型的多个版本，同时也支持监控各个模型版本的一致性，以及并行开发的版本可能产生的合并结果。

然而，模型转换通过一致性关系有效地连接多个模型，仅考虑单个模型的演化而忽略其上下文，对于较大的模型驱动软件开发项目而言是不够的。因此，需要一种能够建立此类相互关联模型的不同版本之间一致性的机制，同时还要支持多个版本的并行开发。

这种机制将支持使用更紧凑的表示形式，同时还能实现 [2]中所述的进一步分析操作。此外，集成处理多个模型版本可能有助于提升转换的执行时间性能。

在本报告中，我们通过调整广为人知的三元图文法形式化体系，提出了朝向适用于多版本模型的模型转换迈出的第一步；该形式化体系支持将单版本模型转换的实现扩展到多版本情形。

本报告的其余部分结构如下：第 2 节简要重述本报告所使用的图、图转换、三元图文法和多版本模型的基本概念。随后，第 3 节以三元图文法的形式，介绍如何从单版本模型转换规范推导出适用于多版本模型的转换规则。第 4 节描述如何使用所推导的规则，实现对多版本模型中编码的所有单独模型版本进行联合转换，并证明我们的方法相对于三元图文法语义的正确性。第 5 节报告对所提出方案性能的初步评估结果，评估内容涉及执行时间，并基于软件开发领域中的一个应用场景。第 6 节讨论相关工作，最后第 7 节总结本报告。

2 预备知识

本节简要概述所需的预备知识，包括图与图转换、三元图文法以及多版本模型。

2.1 图与图转换

我们简要重述 [6] 中定义、且本报告后续内容所需的图、图态射和图变换及其类型化对应概念。

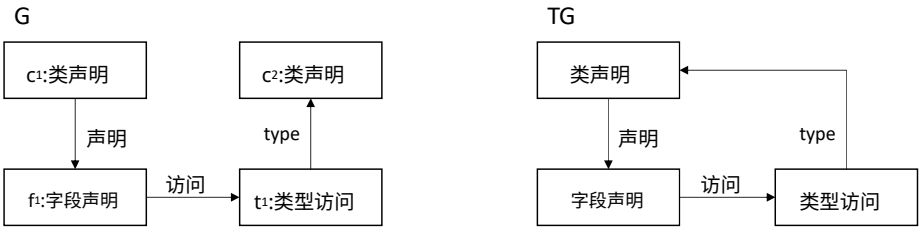


图1：示例图和类型图

图 $G=(V^G, E^G, s^G, t^G)$由节点集 V^G、边集E^G 以及两个函数 s^G：$E^G \to V^G$和 t^G：$E^G \to V^G$组成，分别为每条边指定其源和目标。图态射 m：$G \to H$由一对函数 m^V：$V^G \to V^H$和 m^E：$E^G \to E^H$给出，它们将G中的元素映射到 H中的元素，并满足 $s^H \circ m^E = m^V \circ s^G$和 $t^H \circ m^E = m^V \circ t^G$。我们还称 m^V为顶点态射，称 m^E为边态射。

图 G 可以通过类型态射类型 $G \to TG$映射到类型图 TG 上，从而形成带类型图 $G^T=(G$、类型$^G)$。在本报告中，我们将模型视为带类型图，其中类型图充当元模型，用以定义一种建模语言。

对于两个具有相同类型图的带类型图 $G^T=(G,$ 类型$^G)$ 和$H^T=(H,$ 类型$^H)$，它们之间的类型化图态射表示一个图态射m^T： $G \to H$，满足类型^G = 类型$^H \circ m^T$。图态射（带类型） m 是单态射，当且仅当其函数 m^V和 m^E 均为单射。

图1展示了软件开发领域中的一个带类型图 G 及其对应的类型图 TG。类型态射通过节点的标签编码。 G 表示一个用面向对象编程语言编写的程序的抽象语法图，其中节点可以表示类声明（$ClassDecl$）、字段声明（$FieldDecl$）或类型访问（$TypeAccess$）。类声明可以通过类型为 $declaration$的边包含字段声明；而字段声明可以通过一个 $TypeAccess$ 节点以及类型为 $access$和 $type$的边，将类声明引用为字段类型。示例图包含两个类声明，其中一个包含一个字段声明，该字段的字段类型由另一个类声明指定。

一个（类型化）图变换规则 r 由一对（类型化）图单态射的跨度 $L \xleftarrow{l} K \xrightarrow{r} R$ 刻画，并可通过单态射 m 将其应用于图 G： $L \to G$ 该单态射称为匹配，并满足所谓的悬挂条件 [6]。规则应用所得的结果图 H 随后通过关于中间图 [6]的双重推出作形式化定义。直观而言，应用 r 会删除 m（L）中那些在R 中没有对应元素的元素，并为 R 中那些在 L中没有对应元素的元素创建新元素。图 L 也称为规则的左侧， K 称为规则的粘合图，而 R 称为规则的右侧。

r 如果不删除任何元素，即 l，则称为图产生式

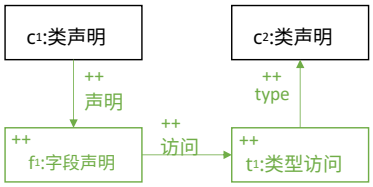


图 2：图变换规则的简写表示法示例

是满射。在这种情况下，由于 L 和 K 通过 l 同构，且我们只区分同构意义下的图，因此也使用简化表示 $L \xrightarrow{r} R$。

图 2 展示了图产生式简写表示法的示例，其中保留的元素以黑色标示，而新创建的元素以绿色标示，并附有额外的“$++$”标签。对于两个已有的类，该产生式会在其中一个类中创建一个字段声明，并将另一个类声明作为字段类型引用。

我们将规则集 R 中的规则在图 G 上依次应用，并得到结果图 G' 的序列记作 $G \rightarrow^{R} G'$。如果无法再应用 R 中的任何规则来延长该规则应用序列，则称该序列为极大规则应用序列。

定义 1. 极大规则应用序列 如果规则集 R 中的（多版本或原始）前向规则所构成的规则应用序列 $G \rightarrow^{R} G'$ 无法再应用 R 中的任何规则，则该序列为极大规则应用序列。

2.2 三元图文法

三元图文法最初由 Schuerr 提出 [19]。本报告基于 [9] 中介绍的稍作调整的版本。

在 [9], 中，三元图文法（TGG）通过一种对应建模语言关联源建模语言和目标建模语言，并由一组 TGG 规则定义。TGG 规则由一个图产生式定义，该产生式同时将源、对应和目标建模语言中的连通图转换为经过一致修改的图三元组。TGG 规则集必须包含一个专门的公理规则，其左侧是由空图组成的三元组，并通过其右侧实际定义一组三元起始图。

TGG规则的左侧 $r = L \xrightarrow{r} R$ 可分别划分为源域、对应域和目标域 L_S、 L_C 和 L_T，其中包含 $L_S \subseteq L$、$L_C \subseteq L$以及 $L_R \subseteq L$ 和 $L_S \uplus L_C \uplus L_R = L$。右侧同样可划分为三个域 R_S、 R_C和 R_T。因此，图三元组和TGG规则的类型图由定义源语言、对应语言和目标语言的类型图的并集构成，并加入额外的边，将对应语言中的节点连接到源语言和目标语言中的节点。

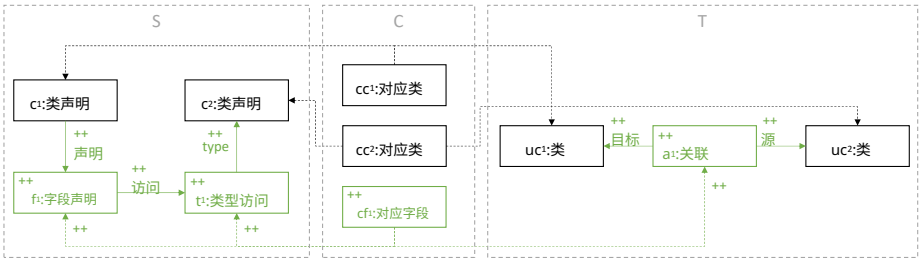


图3：三图语法规则的简写表示法示例

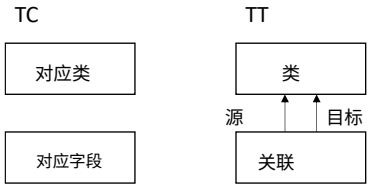


图4：图3中三图语法规则的类型图示例

图3展示了一条TGG规则，用于将图1中的类型图所定义的抽象语法图语言与图4中的类型图 TT 所定义的类图建模语言关联起来，并使用图4中的对应语言 TC 。该规则同时在源域（标记为S）中创建一个 $FieldDecl$ 和$TypeAccess$ 及其关联的边，并在目标域（标记为T）中创建一个相应的 $Association$ 及其关联的边；它们通过对应域（标记为C）中新创建的 $CorrField$类型对应节点相连接。

TGG可用于将源语言的模型转换为目标语言的模型。这需要从TGG规则集合中推导出所谓的前向规则。对于TGG规则 $r = L \xrightarrow{r} R$，可将前向规则构造为 $r^F = L^F \xleftarrow{id} L^F \xrightarrow{r^F} R$，其中 $L^F = L \cup (R_S \setminus r(L))$，且 $r^F = r \cup id$为恒等态射。直观而言， r^F已经要求源域中存在通过应用 r 将创建的元素，并且仅在对应域和目标域中创建元素。在下文中，我们还将前向规则中对应于该规则新转换的子图记为 $L^T = L^F \setminus L$。

此外，前向规则的推导需要进行技术扩展，以避免对同一元素进行重复翻译。因此，引入了一个专用的记账节点，它通过记账边与每个当前尚未翻译的源元素相连。随后，将一个记账节点以及连接到 L^T 中所有元素的记账边添加到前向规则的左侧。记账节点也会被添加到规则的粘合图和右侧。此外，还会将负应用条件添加到 L^F,

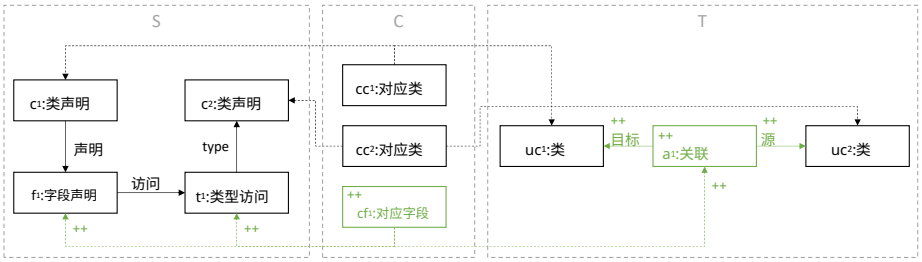


图5：从图3中的三图语法规则派生的示例前向规则，为便于阅读省略了记账机制

这些条件确保对于从 L^F 到 SCT 的匹配 m， $\forall x \in L^F \setminus L^T$： $\nexists b \in B^{SCT}$： $t_B^{SCT} = m(x)$。

因此，通过 m 应用前向规则要求$m(L^T)$中的元素尚未翻译，这一点由记账边的存在表明；同时，通过删除相邻的记账边，将这些元素标记为已翻译。相比之下， $m(L^F \setminus L^T)$中的元素必须已经翻译。请注意，为了允许记账节点与常规边之间存在记账边，本文使用了略有扩展的图模型，详见[10]。

图5展示了从图3中的三图语法规则派生的前向规则。 f_1 和 t_1 及其相邻边不再被创建，而是被保留。此外，该规则要求 f_1、 t_1及其相邻边具有记账边，并包含负应用条件，禁止 c_1 和 c_2存在记账边。不过，为便于阅读，图中省略了这一记账机制。随后，应用该规则会删除指向 f_1、 t_1及其相邻边的记账边，并在目标域中创建相应元素，同时在对应域中创建连接节点 cf_1 。

TGG 也可通过类似推导出的后向规则，将目标语言转换为源语言。下文将重点讨论正向情形。不过，反向情形的处理方式与此类似。

规则之间不存在任何临界对的三图语法 [6] 称为确定性的[9]。使用确定性三图语法进行前向转换时，可以通过操作 $trans^F$来执行：只要存在任何规则的匹配，就持续应用该三图语法的前向规则。由于不存在临界对，规则应用的顺序不会影响最终结果。具体而言，对于包含一组前向规则 R 和起始模型三元组 SCT的确定性三图语法，如果某个极大规则转换序列 $SCT \rightarrow^R SCT'$ 删除了 SCT中的所有记账边，则该序列构成正确的模型转换。请注意，如果$SCT \rightarrow^R SCT'$ 满足此记账判据，那么针对 SCT 和 R 的所有其他可能的极大规则转换序列也满足该记账判据。本报告将重点讨论这类确定性三图语法，它们支持高效的实际实现，从而避免前向规则应用和回溯过程中可能代价高昂的撤销

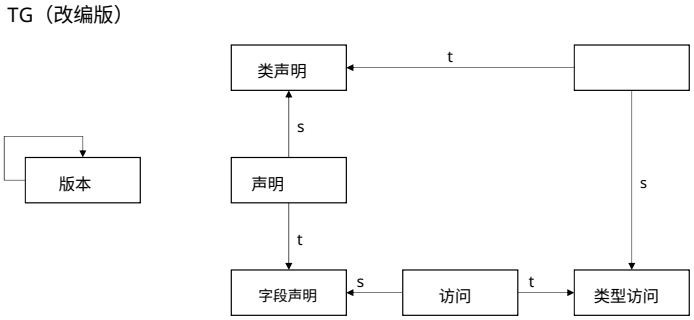


图6：一个改编的示例类型图，源自图1中的类型图；为便于阅读，省略了 cv 和 dv 边

操作 [9]。

2.3 多版本模型

在本报告中，我们考虑以带类型图形式表示的模型。在此语境下，模型修改可以表示为一个态射跨度 $M \leftarrow K \rightarrow M'$，其中 M'是原始模型，它经由中间模型 K被修改为变更后的模型 M'，与图变换步骤 [21]类似。模型的版本历史 由模型 M_1、M_2、.....、M_n之间的一组模型修改 $\Delta^{M_{\{1,\ldots,n\}}}$给出，这些模型具有类型图 TM。我们将具有唯一初始版本，且各个版本之间的模型修改关系无环的版本历史称为正确的版本历史。

在 [2], 中，我们介绍了多版本模型，将此类版本历史编码在单个合并图中。为此，创建了一个改编版本 TM, TM_{mv}。为表示模型结构，TM_{mv}为 TM中的每个节点和每条边都包含一个节点。TM 中边的源和目标关系由 TM_{mv}中的边表示。此外，在 TM_{mv}中添加一个带有自反 suc 边的 $version$ 节点，从而可以物化该版本历史的版本图。版本图与模型结构通过 TM_{mv} 中每个节点 v 到 $version$ 节点的 cv_v 和 dv_v 边相连。

图1中类型图适配后的结果如图6所示。请注意，为了便于阅读，省略了 cv 和 dv 边。

TM_{mv}允许通过过程$comb$将 $\Delta^{M_{\{1,\ldots,n\}}}$ 转换为一个符合 TM_{mv}的单一类型化图 MVM，该图称为多版本模型。由此得到一个双射函数 $origin$：$V^{MVM} \rightarrow \bigcup_{i\in\{1,2,\ldots,n\}} V^{M_i} \cup E^{M_i}$，将 MVM中的顶点映射到各自的原始元素。通过投影操作$proj$(MVM，i)，可以从 MVM中提取单个模型版本 $= M_i$。最后，对于顶点 $v_{mv} \in V^{MVM}$，可通过函数 p计算包含元素 $origin$（v_{mv}）的模型版本集合，其中 p（v_{mv}）$= \{M_i \in \{M_1, M_2, ..., M_n\} | origin$（$v_{mv}$）$\in M_i\}$。

3 从三元图文法推导多版本转换规则

使用三元图文法对多版本模型中编码的各个模型版本进行转换，可以通过投影操作 $proj$轻松实现。然而，与通过 $proj$推导出的所有模型版本的显式枚举相比，多版本模型在实践中可能提供更紧凑的表示。

在这类实际应用场景中，直接作用于多版本模型的、涉及所有模型版本的操作，其执行时间也可能优于针对各个单独模型版本执行的相应操作。我们已经在版本历史 [2]中检查所有模型版本良构性的模式匹配这一情形下证明了这一点。由于模式匹配也是通过三元图文法进行模型转换时的一项重要任务，因此，基于多版本模型表示直接对所有模型版本进行联合转换似乎是可取的。

给定一个三元图文法 TGG，可以直接从常规转换规则推导出图转换规则，用于联合转换编码在多版本模型中的所有源模型版本或目标模型版本。下文将讨论正向转换规则的推导。反向情形的规则也可以用类似的方法推导。

首先，通过 [2]中所述的转换过程，为三图语法合并后的源、对应关系和目标类型图创建适配后的多版本类型图。所得的适配类型图 TG_{mv} 针对多版本模型，为合并类型图源域中的每个节点 v 增加两条边 ucv_v 和 udv_v。这些边的源和目标分别由 $s^{TG_{mv}}(ucv_v) = s^{TG_{mv}}(udv_v) = v$ 以及 $t^{TG_{mv}}(ucv_v) = t^{TG_{mv}}(udv_v) = version$给出，其中 $version$ 是适配类型图中专用的版本节点。

与原始类型图中的记账边类似，这些边将在转换过程中用于编码节点 v_{mv} 所表示的具有类型 v 的元素已经在哪些版本中完成转换。因此，我们类似地定义尚未完成转换 v_{mv} 的版本集合 $u(v_{mv})$，其方式与存在 v_{mv} 的版本集合 $p(v_{mv})$相同，只是在定义中用 ucv_v和 udv_v 替换 cv_v 和 dv_v。

然后，对于每条前向规则 $r = L \xleftarrow{l} K \xrightarrow{r} R$，通过过程 $adapt$创建相应的多版本前向规则，其中 $adapt(r) = trans'(L) \xleftarrow{l_{mv}} trans'(K) \xrightarrow{r_{mv}} trans'(R)$。$l_{mv}$ 的顶点态射由 $l^V_{mv} = origin^{-1} \circ l \circ origin$ 给出，对于表示源关系和目标关系的所有边，其边态射分别由 $l^E_{mv} = s \circ origin^{-1} \circ l^E \circ origin \circ s^{-1}$ 和 $l^E_{mv} = t \circ origin^{-1} \circ l^E \circ origin \circ t^{-1}$ 给出。r_{mv} 以类似方式构造。

$trans'$ 过程是对 [2],中 $trans$ 过程的轻微适配，它忽略了记账节点、记账边和负应用条件，但其他方面的工作方式类似。记账机制则被转换为关于 $trans'(L)$的附加约束 $P \neq \emptyset$，其中$P = (\bigcap_{v_{mv} \in V^{trans'(L)}} p(v_{mv}) \cap \bigcap_{v_{mv} \in origin^{-1}(L^T)} u(v_{mv})) \setminus \bigcup_{v_{mv} \in V^{trans'(L)}} u(v_{mv})$。

适配规则的应用还会为所有顶点 $v_{mv}^{C} \in V^{trans_{(R)}} \backslash (origin^{-1} \circ r \circ origin)($ $trans(K))$创建出向 cv 边和 dv边，以实现赋值 $p(v_{mv}^{C}) := P$。此外，对于 $v_{mv} \in origin^{-1}(r(l^{-1}(L^{T})))$，该规则的应用还会添加和删除出向 ucv 边和 udv 边，以实现修改 $u(v_{mv}) := u(v_{mv}) \setminus P$。

请注意，由于计算 p 和 u 集合需要考虑任意长度的路径，因此从技术上讲，这些计算无法定义为图转换的一部分，而必须在外部实现。

对于前向规则集合 R，相应的多版本前向规则集合只需定义为 $R_{mv} = \{adapt(r)$ $|r \in R\}$。

4 多版本转换的执行

根据指定的三图语法，对多版本模型 MVM 中编码的所有模型版本进行前向转换，可以通过该三图语法的多版本前向规则集合联合完成。

首先，移除 MVM 中存在的所有 ucv 和 udv 边。然后，对于每条满足 $type(e_{cv}) = cv_x$ 和 $s^{MVM}(e_{cv})$ 的边 $e_{cv} \in E^{MVM}$，创建一条满足 $type($ $e_{ucv}) = ucv_x$ 和 $s^{MVM}(e_{cv}) = s^{MVM}(e_{ucv})$ 和 $t^{MVM}(e_{cv}) = t^{MVM}(e_{ucv})$ 的边 e_{ucv}。对于所有 dv 边，以类似方式创建相应的 udv 边。因此，创建 ucv 和 udv 边后，有$\forall v_{vm} \in V^{MVM}$: $u(v_{vm}) = p(v_{vm})$。

随后，对 MVM 中编码的所有模型版本进行的同步转换，采用与通过三图语法对单个模型版本进行常规转换相似的方式执行。具体而言，只需将三图语法的适配后的前向规则应用于 MVM，直到不再有此类规则可应用为止。

接下来，我们将论证，这种转换方法是正确的，因为它产生的结果与通过常规前向规则转换单个模型版本所得的结果相同。

因此，我们将从 [2] 出发的投影操作 $proj$ 扩展为一个簿记敏感变体。

定义 2. *(簿记敏感投影)* 对于具有版本图 V 和版本 $t \in V^{V}$ 的多版本模型 MVM，簿记敏感投影操作的工作方式与常规投影操作 $proj$ 相似，不同之处在于，当且仅当对于所有 $v \in V^{MVM}$， $t \notin u(v)$ 时，它还会向元素 $origin(v)$ 添加一个簿记节点和若干簿记边。我们还将簿记敏感投影操作的结果记为 $MVM[t] = proj^{M}(MVM, t)$。

我们还定义了两个集合，用于表示转换过程中的记账。

定义 3. （记账集）对于模型 M，我们用 $B(M) = \{x \in M | \nexists b \in E'^{M}:$ $t'^{M} = x\}$ 表示已翻译元素（顶点和边）的集合，其中 E'^{M} 是 M 中的记账边集合， t'^{M} 是记账边的目标函数。我们还将 $B(M)$ 称为 M 的记账集。

定义 4。（投影记账集）对于多版本模型 MVM 和版本 $t \in V^V$，其版本图为 V，我们将 $MVM[t]$ 中已处理元素（顶点和边）的集合记为 $B_{mv}(MVM[t]) = \{x \in MVM[t] | t \notin u(proj^{-1}(x))\}$。我们也将 $B_{mv}(MVM[t])$ 称为 $MVM[t]$ 的投影记账集。

以下定理指出，在通过适配后的前向规则启动转换过程时，经记账敏感投影处理的预备多版本模型能够正确编码各个模型版本翻译工作的初始状态。

定理 1。给定一个多版本模型 MVM，它编码了包含模型版本 M_1、M_2、.....、M_n 的版本历史，且满足 $\forall v_{vm} \in V^{MVM}$：$u(v_{vm}) = p(v_{vm})$，则有 $\forall t \in \{1, 2, ..., n\}$：$MVM[t] = init_F(M_t)$（相差一个同构），其中 $init_F(SCT_t)$ 表示在为常规前向转换过程准备 M_t 时产生的带有记账信息的图，也就是在图 M_t 上添加一个记账节点，并为 M_t 中的所有元素添加记账边所得的图。

证明。直接由以下事实得出：$\forall t \in \{1, 2, ..., n\}$：$proj(MVM, t) = M_t$，这一点已在 [2], 中证明；再结合簿记敏感投影操作的定义即可。

□

根据定理 1，我们还可以得到以下推论：

推论 1。给定一个多版本模型 MVM，它编码了一个包含模型版本 M_1，M_2，.....，M_n 的版本历史，且满足 $\forall v_{vm} \in V^{MVM}$：$u(v_{vm}) = p(v_{vm})$，那么在同构意义下有 $\forall t \in \{1, 2, ..., n\}$：$B_{mv}(MVM[t]) = B(init_F(M_t))$。其中，$init_F(SCT_t)$表示为常规前向转换过程准备 M_t 后得到的带记账的图，即在图 M_t 中添加一个记账节点，并添加连接到 M_t中所有元素的记账边。

证明。直接由定理 1 以及簿记集和投影簿记集的定义得出。

□

现在我们证明，当且仅当某个多版本规则应用所影响的所有单个模型版本均适用相应的常规规则时，该多版本规则才适用于多版本模型。

定理 2。多版本前向规则 $r_{mv} = L_{mv} \leftarrow K_{mv} \rightarrow R_{mv}$ 可通过匹配 m应用于带有簿记的多版本模型三元组 SCT_{mv}，当且仅当对于所有 $t \in P$，相应的原始前向规则 $r = L \leftarrow K \rightarrow R$可通过匹配 $trans(m)$应用于 $SCT_{mv}[t]$，且 $P = \bigcap_{v \in V^{L_{mv}}} p(m(v)) \cap \bigcap_{v \in V^{L^T_{mv}}} u(m(v))$。

证明。对于一个版本 t，正如我们在 [2], 中已经证明的那样，匹配 $m : L_{mv} \rightarrow SCT_{mv}$ 存在一个对应的匹配 $trans(m)$：$L \rightarrow SCT_{mv}[t]$，当且仅当

$t \in \bigcap_{v \in V^{L_{mv}}} p(m(v))$。此外，根据 P的定义和 r_{mv}的构造， $m(trans(m)(L^T))$中的所有元素在 $SCT_{mv}[t]$ 中都有一条相邻的记账边，当且仅当 $t \in \bigcap_{v \in V^{L^T_{mv}}} u(m(v))$。类似地， $m(trans(m)(L \setminus L^T))$中的所有元素在 $SCT_{mv}[t]$ 中都没有相邻的记账边，当且仅当$t \notin \bigcup_{v \in V^{L_{mv} \setminus L^T_{mv}}} u(m(v))$。由于 r 和 r_{mv} 均未删除顶点，因此 r 和匹配 $trans(m)$显然满足悬挂条件。因此， r_{mv} 可通过 m应用于 SCT_{mv} ，且 $t \in P$成立，当且仅当 r 可通过$trans(m)$应用于 $SCT_{mv}[t]$ 。 □

现在我们可以证明，对多版本模型的一次多版本规则应用，等价于将相应的常规规则应用于所有受影响的模型版本。

定理 3. 对于通过匹配 m，将带有原始前向规则 $r = L \leftarrow K \rightarrow R$ 的多版本前向规则 $r_{mv} = L_{mv} \leftarrow K_{mv} \rightarrow R_{mv}$ 应用 $SCT_{mv} \rightarrow^{r_{mv}}_{m} SCT'_{mv}$ 于具有记账和版本图 V 的多版本模型三元组 SCT_{mv} ，有 $\forall t \in P$：$SCT'_{mv}[t] = SCT' \wedge B_{mv}(SCT'_{mv}[t]) = B(SCT')$，两者在同构意义下相同，相应的应用为 $SCT_{mv}[t] \rightarrow^{r}_{trans(m)} SCT'$。此外， $\forall t \in V^V \setminus P$: $SCT'_{mv}[t] = SCT_{mv}[t] \wedge B_{mv}(SCT'_{mv}[t]) = B(SCT_{mv}[t])$，两者在同构意义下相同，其中 $P = \bigcap_{v \in V^{L_{mv}}} p(m(v)) \cap \bigcap_{v \in V^{L^T_{mv}}} u(m(v))$。

证明。忽略记账边后，所有前向规则以及适配后的前向规则都是产生式。由于适配后的前向规则的构造方式，该规则应用所创建的所有元素仅在版本 P 中对 SCT'_{mv} 多版本存在。因此，对于所有其余版本， $SCT_{mv}[t]$ 包含的元素与 $SCT'_{mv}[t]$ 相同。在这种情况下，同构 iso： $SCT_{mv}[t] \rightarrow SCT'_{mv}[t]$ 显然由恒等映射给出。由于 r_{mv} 的应用只会改变版本 P 的投影记账集，$B_{mv}(SCT'_{mv}[t]) = B(SCT_{mv}[t])$，其中同构为 iso。

因此，在同构意义下有： $\forall t \in V^V \setminus P$: $SCT'_{mv}[t] = SCT_{mv}[t] \wedge B_{mv}(SCT'_{mv}[t]) = B(SCT_{mv}[t])$。

r_{mv} 到 SCT_{mv} 的应用产生了一个余匹配 n： $R_{mv} \rightarrow SCT'_{mv}$；类似地，$r$ 到 $SCT_{mv}[t]$ 的应用也为任意 $t \in P$产生了一个余匹配 n'： $R \rightarrow SCT'$。

随后可按如下方式构造一个同构 iso： $SCT'_{mv}[t] \rightarrow SCT'$ 。由于 r_{mv} 是一个产生式， SCT_{mv} 是 SCT'_{mv} 的子图，因此 $SCT_{mv}[t]$也是 $SCT'_{mv}[t]$的子图。由于 r 是一个产生式， $SCT_{mv}[t]$ 也是 SCT'的子图。因此， $SCT'_{mv}[t]$ 与 SCT' 之间针对 $SCT_{mv}[t]$ 的同构映射只需由恒等映射给出。这样，只有n（$R_{mv} \setminus L_{mv}$）中的元素和 n'（$R \setminus L$）中的元素尚未映射。由于 r_{mv} 的构造在同构意义下唯一， n 和 n' 是单态射，且$trans$ 和 $origin$ 是双射，其余同构映射由 $n' \circ trans \circ n^{-1} \circ origin$给出。注意，对于 n（L_{mv}）中的元素，通过恒等映射和 $n' \circ trans \circ n^{-1} \circ origin$ 来定义iso 是多余的，但两者兼容。

根据簿记敏感投影、簿记集和投影记账集的定义，有 $B(SCT_{mv}[t]) = B_{mv}(SCT_{mv}[t])$

因此 $B_{mv}(SCT_{mv}[t]) = B(SCT_{mv}[t])$. 与 $B_{mv}(SCT_{mv}[t])$ 相比， r_{mv} 的应用仅通过添加 $trans(m(L^T_{mv}))$ 中的元素来更改投影记账集 $B_{mv}(SCT'_{mv}[t])$。因此，对 $B_{mv}(SCT'_{mv}[t])$ 的修改对应于通过 $trans(m)$ 对同构 iso 应用 r 时对簿记集 $B(SCT')$ 的修改，这是由 r_{mv} 的构造所致。

因此有 $\forall t \in P$： $SCT'_{mv}[t] = SCT' \wedge B_{mv}(SCT'_{mv}[t]) = B(SCT')$。

□

基于针对单个规则应用的定理 3，我们可以得到以下关于规则应用序列的推论:

推论 2。对于一个三图语法，其关联的正向规则集合为 R 、多版本正向规则为 R_{mv} ，以及带有簿记和版本图 V 的多版本模型三元组 SCT_{mv} ，存在规则应用序列 $SCT_{mv} \rightarrow^{R_{mv}} SCT'_{mv}$ ，当且仅当对于所有 $t \in V^V$，存在规则应用序列 $SCT_{mv}[t] \rightarrow^{R} SCT'$ ，且有 $SCT'_{mv}[t] = SCT' \wedge iso(B_{mv}(SCT'_{mv}[t])) = B(SCT')$, 其中 iso 是从 $SCT'_{mv}[t]$ 到 SCT' 的同构。

证明。我们通过对多版本规则应用序列的长度进行归纳，证明该推论。

对于长度为 0 的应用序列，恒等态射和空应用序列显然满足该推论。

存在规则应用序列 $SCT_{mv} \rightarrow^{R_{mv}} SCT'_{mv}$ 当且仅当对于所有 $t \in V^V$，都存在规则应用序列 $SCT_{mv}[t] \rightarrow^{R} SCT'$，满足 $SCT'_{mv}[t] = SCT' \wedge iso(B_{mv}(SCT'_{mv}[t])) = B(SCT')$；根据定理 3，我们有一个扩展多版本序列 $SCT_{mv} \rightarrow^{R_{mv}} SCT'_{mv} \rightarrow^{r_{mv}}_{m} SCT''_{mv}$，并且所有 $t \in V^V$ 当且仅当对于所有 $t \in V^V$，都存在常规规则应用序列 $SCT_{mv}[t] \rightarrow^{R} SCT''$，满足 $SCT''_{mv}[t] = SCT'' \wedge iso(B_{mv}(SCT''_{mv}[t])) = B(SCT'')$。

对于所有 $t \in V^V \setminus P$，其中 $P = \bigcap_{v \in V^{L_{mv}}} p(m(v)) \cap \bigcap_{v \in V^{L^T_{mv}}} u(m(v))$, 相应的常规规则应用序列 $SCT_{mv}[t] \rightarrow^{R} SCT'$ 和同构 iso：
$SCT'_{mv}[t] \rightarrow SCT'$ 对 $SCT''_{mv}[t]$ 也有效，并满足关于记账集合的条件，因为 $SCT' = SCT'_{mv}[t] = SCT''_{mv}[t]$ （在同构意义下）。

根据定理 3，当且仅当对于所有 $t \in P$，常规规则应用序列 $SCT_{mv}[t] \rightarrow^{R} SCT'_{mv}[t]$ 可以通过一个满足关于记账集合条件的规则应用$SCT'_{mv}[t] \rightarrow^{r}_{trans(m)} SCT''_{mv}[t]$ 进行扩展时，才存在一个扩展序列 $SCT_{mv} \rightarrow^{R_{mv}} SCT'_{mv} \rightarrow^{r_{mv}}_{m} SCT''_{mv}$。

因此，存在一系列规则应用 $SCT_{mv} \rightarrow^{R_{mv}} SCT'_{mv} \rightarrow^{r_{mv}}_{m} SCT''_{mv}$ 当且仅当对于所有 $t \in V^V$，都存在一系列规则应用 $SCT_{mv}[t] \rightarrow^{R} SCT''$，使得 $SCT''_{mv}[t] = SCT'' \wedge iso(B_{mv}(SCT''_{mv}[t])) = B(SCT'')$。

结合基本情形的证明和归纳步骤，我们已经证明了该推论的有效性。

□

直观而言，多版本正向规则对编码在 SCT_{mv}中的所有模型版本执行交错式并行转换。多版本规则 $L_{mv} \leftarrow K_{mv} \rightarrow R_{mv}$ 的应用对应于将原始规则应用于 $P = \bigcap_{v \in V^{L_{mv}}} p(m(v)) \cap \bigcap_{v \in V^{L^T_{mv}}} u(m(v))$ 中的所有模型版本，并保持其他所有模型版本不变。因此，多版本规则应用实际上通过相应的原始规则应用，扩展了 P 中所有版本对应的原始规则应用序列；而对于不在 P 中的版本，则表示跳过其序列中的一步。

对于确定性三图语法，源图 S 的正确转换由任意一个能够删除源模型中所有记账边的正向规则极大规则应用序列给出。请注意，由于确定性判据，要么每个极大规则应用序列都满足记账判据，要么都不满足。因此，通过多版本前向规则对所有各个版本进行联合转换的正确性可由以下推论给出:

推论 3。对于一个三图语法，其关联的正向规则集合为 R 、多版本前向规则集合为 R_{mv}，且给定一个包含记账信息和版本图 V 的多版本模型三元组 SCT_{mv}，当且仅当对于所有 $t \in V^V$，都存在一个极大常规规则应用序列 $SCT_{mv}[t] \rightarrow^R SCT'$，使得 $SCT'_{mv}[t] = SCT' \wedge B_{mv}(SCT'_{mv}[t]) = B(SCT')$ 时，存在一个极大规则应用序列$SCT_{mv} \rightarrow^{R_{mv}} SCT'_{mv}$。

证明。推论 2 表明，对于一系列多版本规则应用，以及所有版本 $t \in V^V$，都存在与之对应的一系列原始规则应用，反之亦然。由定理 2 可直接得出：当且仅当所有 $t \in V^V$ 的常规序列均为极大序列时，多版本序列才是极大序列。 □

因此，对于确定性三图语法，根据推论 1 和推论 3，将改编的转换规则反复应用于为多版本转换准备的多版本模型，直至达到不动点，其结果等价于将原始规则反复应用于为转换准备的各个模型版本所得的结果；换言之，这等价于使用三图语法转换各个模型版本所得的结果。

由此，我们证明了使用多版本正向规则 $trans^F_{mv}$ 进行前向转换的正确性：该过程将多版本正向规则应用于带有记账信息的多版本模型，直至达到不动点。

定理 4。对于正确的版本历史记录 $\Delta^{M_{\{1,\dots,n\}}}$ 和一组三图语法前向规则 R，在同构意义下有

$$\forall t \in \{1, ..., n\} : trans^F_{mv}(init_F(comb(\Delta^{M_{\{1,\dots,n\}}})), adapt(R))[t] = trans^F(M_t, R) \tag{1}$$

证明。由定理 1 和推论 3 直接得出。 □

5 评估

为了从执行时间性能方面对我们的方法进行实证评估，我们已在用于基于TGG的模型转换的 MoTE2 工具中实现了所介绍的概念 [12] 。该工具基于 Java 的 Eclipse建模框架实现 [7] ，并且已证明，与其他现有模型转换工具相比，它具有较高的效率 [12]。

作为一个应用场景，我们考虑将 Java 抽象语法树转换为类图。因此，我们使用 MoTE2 将这一转换建模为三图语法（TGG），并分别使用原始实现和我们改编后的实现，自动推导所需的前向规则和多版本前向规则。

为了获得真实的源模型，我们从各自的 GitHub 仓库中提取了一个小型个人 Java 项目（$rete$，约 50 个版本）和一个较大型开源 Java 项目（$henshin$ [1], 约 2000 个版本）的版本历史，并使用 MoDisco 工具 [3]构建了相应的抽象语法树历史。作为第 3 节和第 4 节所述解决方案的输入，我们基于层次结构和命名映射，将这两段版本历史整合为多版本模型。

我们的实现和所用数据集可通过 [22] 获取。
基于此，我们对两个仓库分别运行以下模型转换，并测量每项转换的总体执行时间[1]：

- **单版本模型**：使用原始 MoTE2 实现，对仓库中的所有模型版本（抽象语法树）分别执行前向转换
- **多版本模型**：使用多版本模型编码以及我们对第3节和第4节所述技术的实现，对仓库中的所有模型版本进行联合前向转换

请注意，单版本模型策略需要先执行投影操作，最后再组合转换结果，才能在多版本模型框架中运行。不过，为公平比较转换效果，我们在评估中不考虑这些额外操作。

图7展示了采用这两种策略进行转换的执行时间。对于两个仓库，基于多版本模型的转换所需时间都显著少于使用原始 MoTE2 工具转换各个单独模型版本的时间；在较大的仓库中，提升更加明显（较小仓库的加速因子为4，较大仓库为74）。

效率和可扩展性的提升可能可以用两个因素来解释：首先，对于每个需要转换的单独模型版本，SVM 都必须执行一个开销较大的初始化步骤，而 MVM 只

[1]所有实验均在一台运行 Linux SMP Debian 4.19.67-2 的机器上进行，该机器配备 Intel Xeon E5-2630 CPU（时钟频率为 2.3 GHz）和 386GB 系统内存，运行 OpenJDK 11.0.6 版。报告的执行时间测量值为相应实验运行10次的平均执行时间。

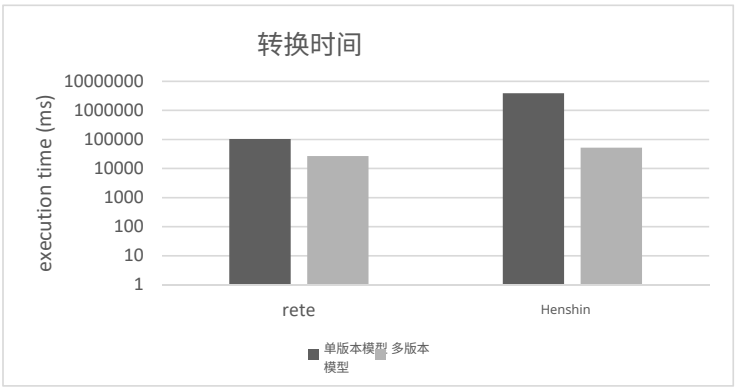


图7：在两个不同的软件仓库中转换所有模型版本的执行时间测量结果（对数坐标轴）

需执行一次这样的初始化。其次，仓库中抽象语法树的许多元素在多个版本之间共享。SVM 必须针对每个模型版本分别执行转换，包括单独进行模式匹配。相比之下，MVM 只需对一个多版本模型执行转换并进行模式匹配；该模型的规模远小于各个编码后模型版本的总规模，同时还可在版本图上执行高效的搜索操作。由于在此示例中模式匹配效率很高，也就是说，对于派生出的前向规则，模式匹配的运行时复杂度与模型规模呈线性关系，因此整体效率得以提升。

实验结果的内部效度可能受到 Java 虚拟机的意外行为（如垃圾回收）的威胁。为应对这一威胁，我们对所有实验进行了多次运行，并报告这些运行的平均执行时间；执行时间的标准差始终低于 5%。为尽量减少具体实现对测量结果的影响，我们在用于比较的转换工具框架中实现了自己的解决方案，因此基本采用了相同的执行机制。

为减轻外部效度方面的威胁，我们使用真实世界模型作为转换的源模型。不过，我们指出，我们的结果未必能推广到不同的示例或应用领域，也不对我们方法的性能作出任何定量声明。

6 相关工作

模型版本管理这一普遍问题已得到广泛研究，既包括形式化研究 [5, 17] ，也包括具体工具的实现 [14, 13]。一些解决方案采用统一表示来呈现模型的版本历史

这与多版本模型类似 [17, 14]。然而，由于问题定义侧重于管理单个模型的不同版本，因此这些方法并未涉及基于统一编码实现模型转换。

此外，还有大量关于使用三元图文法 [24, 15]同步并发修改的模型对的既有研究。这些工作的重点是推导出兼容的源模型和目标模型版本，使其尊重对任一模型所做的修改。本报告旨在朝一个正交的方向迈出一步，即通过允许开发者暂时使用多个经过修改且可能相互冲突的源模型和目标模型版本，使其能够与不一致性共存。

在软件产品线领域，人们使用所谓的 150%模型来编码软件系统 [4, 16]的不同配置。在此背景下，Greiner 和 Westfechtel 提出了一种方法，用于沿着模型转换 [23],所创建的追踪链接传播所谓的可变性注释，并明确考虑了通过三元图文法实现转换的情形。类似的方法也可用于传播版本信息，其优点是不需要对所用规则、类型图或转换过程进行任何适配。然而，如果不将这种传播与转换过程集成，而只在转换执行完毕后传播版本信息，就意味着我们的方法能够处理的某些情形将无法处理。因此，这些情形的出现可能会妨碍正确转换。例如，在标准三元图文法语义下，这类情形包括：同一模型元素根据其上下文，在不同模型版本中被转换为不同形式。

我们小组此前的工作研究了如何在演变中的模型的多个版本上联合执行查询，既考虑了存在 [2] 并行分支式开发的情况，也考虑了不存在 [11, 18]并行分支式开发的情况。本报告以这些成果为基础，但不再关注没有副作用的纯查询，而是研究以模型转换形式进行写操作的情况。

7 结论

在本报告中，我们迈出了多版本模型模型转换研究的第一步：通过适配广为人知的三元图文法形式体系，实现对多版本模型中编码的所有版本进行联合转换。就针对单个模型版本的确定性三元图文法的转换语义而言，所提出的方法是正确的，也就是说，它会产生等价的结果。为评估该方法效率而开展的初步实验表明，在真实的应用场景中，与朴素实现相比，我们的技术能够提升性能；朴素实现会根据三元图文法规范，分别转换所有模型版本。

在未来的工作中，我们计划在所提出的方法基础上，实现多版本模型的同步，即将对源模型一个或多个版本的

更改增量传播到相应的目标模型版本。此外，我们希望探索通过对多版本模型进行增量模式匹配来提高多版本模型转换效率的可能性。未来工作的另一个有趣方向是将高级应用条件集成到我们的方法中，用于指定三元图文法规则，例如嵌套图条件。最后，可以开展更全面的评估，以更详细地研究所提出技术的可扩展性。

致谢

本工作主要在“模块化与增量式全局模型管理”项目（项目编号336677879）期间开展，该项目由德国研究联合会资助。

参考文献

[1] Thorsten Arendt 等。“Henshin：用于原位 EMF模型转换的高级概念与工具”。载于：模型驱动工程语言与系统国际会议。Springer。2010年，第121–135页。[2] Matthias Barkowsky 和 HolgerGiese。“迈向多版本模型开发：检测合并冲突并检查良构性”。载于：图转换。Nicolas Behr 和 Daniel Str¨uber 编。Cham：Springer International Publishing，2022年，第118–136页。[3] Hugo Bruneliere 等。“MoDisco：用于模型驱动逆向工程的通用、可扩展框架”。载于：*IEEE/ACM*自动化软件工程国际会议论文集。2010年。 DOI: 10.1145/1858996.1859032。[4] Krzysztof Czarnecki 和 Micha-l Antkiewicz。“将特征映射到模型：基于叠加变体的模板方法”。载于：生成式编程与组件工程。Robert Gl¨uck 和 Michael Lowry 编。第3676卷。LNPSE。柏林、海德堡：Springer Berlin Heidelberg，2005年，第422–437页。 ISBN: 978-3-540-31977-1。 DOI: 10.1007/11561347_28.[5] Zinovy Diskin、Krzysztof Czarnecki 和Michal Antkiewicz。“模型版本管理

-

大规模：代数基础与瓦片表示法”。载于：*2009*年*ICSE*软件模型比较与版本控制研讨会。IEEE。2009年，第7–12页。 DOI: 10.1109/CVSM.2009.5071715。[6] Hartmut Ehrig 等。《代数图转换基础》。EATCS。Springer，2006年。 DOI: 10.1007/3-540-31188-2。[7] *EMF*。https : / / www . eclipse .org / modeling / emf/。最后访问于2022年10月14日。

[8] Anthony C. W. Finkelstein 等。“多视角规范中的不一致处理”。载于：*IEEE*软件工程汇刊20.8（1994），第569–578页。 DOI：10.1109/32.3106670。[9] Holger Giese、Stephan Hildebrandt 和 Leen Lambers。“弥合三元图文法形式语义与实现之间的差距——确保关系模型转换规范与实现的符合性”。载于：软件与系统建模 13.1（2014），第273–299页。 URL：http://dx.doi.org/10.1007/s10270-012-0247-y。[10] Holger Giese、Stephan Hildebrandt 和 Leen Lambers。“迈向弥合三元图文法形式语义与实现之间的差距”。载于：波茨坦大学哈索·普拉特纳研究所数字工程技术报告（37）（2010）。[11] Holger Giese 等。“类型化属性图上的度量时序图逻辑”。载于：软件工程基础方法国际会议。第11424卷。LNTCS。Springer，Cham。2019年，第282–298页。 DOI：10.1007/978-3-030-16722-6_16。[12] Stephan Hildebrandt。《三元图文法实现的性能与符合性》。博士论文。波茨坦大学哈索·普拉特纳研究所，2014年6月。[13] Maximilian Koegel 和 Jonas Helming。“EMFStore：EMF模型的模型仓库”。载于：第*32*届*ACM/IEEE*软件工程国际会议论文集——第*2*卷。2010年，第307–308页。 DOI：10.1145/1810295.1810364。[14] Leonardo Murta 等。“迈向Odyssey-VCS 2：基于UML的版本控制系统的改进”。载于：*2008*年软件模型比较与版本控制国际研讨会论文集。2008年。DOI：10.1145/1370152.1370159。[15] Fernando Orejas、Elvira Pino 和 Marisa Navarro。“使用三元图文法进行增量并发模型同步”。载于：*FASE*。2020年，第273–293页。[16] Dennis Reuling 等。“迈向基于模型的软件产品线的投影式编辑”。载于：第*14*届软件密集型系统可变性建模国际工作会议论文集。VAMOS ’20。德国马格德堡：计算机协会，2020年。 DOI：10.1145/3377024.3377030。[17] Adrian Rutle 等。“一种在模型驱动工程中形式化版本控制的范畴论方法”。载于：软件工程基础方法国际会议。第5503卷。LNTCS。Springer。2009年，第64–78页。 DOI：10.1007/978-3-642-00593-0_5。[18] Lucas Sakizloglou 等。“在运行时模型上增量执行带历史记录的时序图查询及其应用”。载于：软件与系统建模 （2021），第1–41页。 DOI：10.1007/s10270-021-00950-6。

[19] Andy Sch¨urr. “使用三图语法规范图转换器”。载于：《 20^th 国际计算机科学图论概念研讨会论文集。E. W. Mayr、G. Schmidt 和 G. Tinhofer 编。第903卷。《计算机科学讲义》。德国赫尔兴：Spinger Verlag，1994年6月，第151–163页。[20] Andreas Seibel、Stefan Neumann 和 Holger Giese。“动态分层巨型模型：全面的可追溯性及其高效维护”。载于：《软件与系统建模》9.4（2010），第493–528页。[21] Gabriele Taentzer 等。“一种基于图修改的模型版本管理基础方法：从理论到实现”。载于：《软件与系统建模》13.1（2014），第239–272页。 DOI：10.1007/s10270- 012-0248-x。[22] 多版本模型三元图文法评估工件。https://github. com / hpi - sam / TGGs - for - Multi-Version - Models。最后访问于2022年10月14日。[23] Bernhard Westfechtel 和 Sandra Greiner。“通过基于跟踪的可变性注释传播，将单变体模型转换扩展为多变体模型转换”。载于：软件与系统建模19.4（2020），第853–888页。 DOI：10.1007/s10270-020-00791-9。[24] Yingfei Xiong 等。“基于双向转换同步并发模型更新”。载于：《软件与系统建模》12.1（2013）

，

第 89–104 页。