arXiv:2602.15409v1 [cs.LO] 17 Feb 2026

CSLib（Lean计算机科学库）中的 Hennessy–Milner 逻辑

FabrizioMontesi[1][0000−0003−4666−901X]、Marco Peressotti[1][0000−0002−0243−0480] 和 Alexandre Rademaker[2][0000−0002−7583−0792]

[1] FORM，南丹麦大学，欧登塞，丹麦 {fmontesi,peressotti}@imada.sdu.dk[2] 应用数学学院，瓦加斯基金会，里约热内卢，巴西 alexandre.rademaker@fgv.br

摘要。 我们为 Lean计算机科学库（CSLib）实现了 Hennessy–Milner 逻辑（HML）的库级形式化。Hennessy–Milner 逻辑是标记迁移系统（LTS）的基础逻辑。我们的开发内容包括 HML 的语法、满足关系和指称语义，以及完整的元理论，其中包括 Hennessy–Milner 定理——对于像有限标号迁移系统，互模拟等价与理论等价相一致。我们的开发强调通用性和可复用性：它参数化于任意标号迁移系统，定义与 CSLib 的基础设施（例如互模拟等价的形式化）相集成，并利用 Lean 的自动化能力（特别是 grind 策略）完成证明。所有代码均已在 CSLib 中公开，并可直接应用于使用其标号转移系统 API 的系统。

关键词： Hennessy–Milner 逻辑 · 迁移系统 · 互模拟

1 引言

标记迁移系统（LTS）是定义许多计算系统操作行为的首选框架，包括进程演算、通信协议、自动机和反应式系统 [10]。Hennessy–Milner 逻辑（HML）[5] 是一种用于推理此类系统的基础模态逻辑。它的显著特点是*Hennessy–Milner 定理*：对于图像有限的标记迁移系统，两个状态满足完全相同的 HML 命题，当且仅当它们强双模拟等价。这种紧密对应关系使 HML 成为许多场景中非常有用的逻辑基础，从并发理论到模型检查。

尽管 HML 地位举足轻重，但以库级通用性对其进行机械化处理的工作仍然鲜见。现有形式化工作往往针对 HML 的变体或特定类型的标记迁移系统。本文介绍了在 Lean 中开发的一种新的 HML 形式化工作，它是 Lean计算机科学库（CSLib）的一部分。CSLib 是一个快速发展的社区项目，旨在为形式化计算机科学研究和经过验证的软件开发提供通用基础 [2]。

我们的主要贡献包括：(1) HML 的语法、满足关系和指称语义的宇宙多态定义；(2) 完备的

HML 基础元理论，包括其指称语义的正确性以及 Hennessy–Milner 定理（对于像有限标记迁移系统，互模拟等价与理论等价一致）；(3) 将其集成到 CSLib 中，并提供文档和适当的 API，供下游使用。例如，我们的开发可直接应用于自动机理论和 CSLib 中定义的通信系统演算（CCS） [8] ——它们都基于相同的通用标号转移系统 API。

在下文中，我们假设读者对定理证明器有一定了解。代码与可复现性。我们的开发成果公开存放在 CSLib 官方代码库中。[3] 所有代码均使用 Lean 4.28.0-rc1 版本进行了检查。

2 背景：CSLib 中的标记迁移系统与互模拟等价

CSLib 提供了丰富的操作语义库。我们回顾与本文开发工作相关的几个重要要素。

定义 1（标记迁移系统）。一个标记迁移系统是一个三元组 $(S, L, \to)$，其中 S 是状态集合，L 是迁移标签集合，而 $\to \subseteq S \times L \times S$ 是迁移关系。我们用 $s \xrightarrow{\mu} s'$ 表示 $(s, \mu, s') \in \to$。

在 CSLib 中，这被形式化为一个结构，其参数是状态类型和标签类型，`LTS State Label`，其中将迁移关系表示为谓词`Tr : State → Label → State →` 命题。（Lean 代码使用等宽字体书写。）

定义 2。状态 s 在迁移标签 μ 下的像，是所有能够通过 μ-迁移从 s 到达的状态组成的集合：像$(s, \mu) = \{s' \mid s \xrightarrow{\mu} s'\}$。

得益于 Mathlib（Lean 数学库，CSLib 基于它构建）提供的集合记法，CSLib 中的定义与此完全相同 [3] 。

```
def LTS.image (lts : LTS StateLabel) (s : 状态) (μ : 标签) : SetState := { s' : 状态 | lts.Tr s μ s' }
```

定义 3（像有限性）。如果对于每个状态 s和标签 μ，集合 像(s, μ) 都是有限的，那么该标号迁移系统就是像有限的。

在 CSLib 中，这通过为 标号迁移系统定义一个类型类来表达，并要求 `∀ s μ,Finite (lts.image sμ)`。

定义 4（互模拟与互模拟等价）。关系 $R \subseteq S \times S$ 是互模拟，如果每当 $s_1 \, R \, s_2$:

1. $s_1 \xrightarrow{\mu} s_1'$ 对任意 s_1' 都蕴含 $s_2 \xrightarrow{\mu} s_2'$ 存在某个 s_2' 使得 $s_1' \, R \, s_2'$;
2. $s_2 \xrightarrow{\mu} s_2'$ 对任意 s_2' 都蕴含 $s_1 \xrightarrow{\mu} s_1'$ 存在某个 s_1' 使得 $s_1' \, R \, s_2'$。

两个状态 s_1 和 s_2 是互模拟等价的，记作 $s_1 \sim s_2$，当且仅当存在一个互模拟 R使得 $s_1 \, R \, s_2$。

在 CSLib 中，关系 `r : State → State → Prop` 是标号迁移系统 `lts`中的一个互模拟，当且仅当它满足谓词 `lts.IsBisimulation r`。该谓词和互模拟等价是对上述定义的直接形式化。

[3] github.com/leanprover/cslib，撰写时的命名空间为 Cslib.Logic.HML。

3 Lean 中的 HML形式化

我们现在描述 HML 形式化的核心组成部分：命题语法、满足关系（以迁移表述的语义）、指称语义以及理论等价。

3.1 命题

给定一组迁移标签 L，HML 命题语言由以下语法给出：

$$\varphi ::= \text{true} \mid \text{false} \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \langle \mu \rangle \varphi \mid [\mu]\varphi \qquad \text{where } \mu \in L.$$

命题描述给定状态的性质。模态 $\langle \mu \rangle \varphi$ （‘菱形’）断言存在一条从给定状态出发、标签为 μ 、通向满足 φ 的状态的迁移；而 $[\mu]\varphi$ （‘方框模态’）则要求从给定状态出发的所有标签为 μ 的迁移都通向满足 φ 的状态。

我们在 Lean 中将命题定义为一个以标签类型（标签，对应上文的 L ）为参数的归纳类型。

```
归纳定义 命题 (标签： 类型u) ： 类型 u 其中
  | 真  |假命题| 且 (φ₁ φ₂ ：命题 标签)  | 或 (φ₁ φ₂ ：命题标签)| 菱形模态
  (μ : 标签) (φ: 命题 标签)| 方框模态 (μ: 标签) (φ ：命题 标签)
```

该定义是宇宙多态的（u），因此适用于任何标签类型。

HML 中的命题语言有两种常见的处理方式。原始方式不包含假命题，而是包含一个否定联结词 [5]。我们采用另一种来自后续发展的方式，它不包含否定，但包含假命题 [1]。这样一来，满足关系的定义就成为一种优雅、直接的递归（见下一节）。照例，我们可以将否定定义为一个函数，把任意命题转换为其否定形式。

```
定义 命题.否定 (φ ：命题 标签) : 命题 标签 :=匹配  φ 与 | .真  => .假命题  | 且
 φ₁ φ₂ => 或  φ₁.否定  φ₂.否定| 菱形模态μ  φ => 方框模态 μ  φ.否定  | /- ... -/
```

3.2 满足关系

满足关系 $s \models \varphi$ （“状态 s满足命题 φ”）按如下方式归纳定义。假命题没有引入规则，因此没有状态满足它。

$s \models$ 真命题	(没有假命题的规则)
$s \models \varphi_1 \wedge \varphi_2$	如果 $s \models \varphi_1$ 且 $s \models \varphi_2$
$s \models \varphi_1 \vee \varphi_2$	如果 $s \models \varphi_1$ 或 $s \models \varphi_2$
$s \models \langle \mu \rangle \varphi$	如果 $\exists s'. s \xrightarrow{\mu} s' \wedge s' \models \varphi$
$s \models [\mu]\varphi$	如果 $\forall s'. s \xrightarrow{\mu} s' \Rightarrow s' \models \varphi$

同样，我们将其形式化为一个归纳定义。

```
归纳定义 满足关系 (lts : 标号迁移系统 状态 标签) :
    状态 → 命题 标签 → 命题其中| true : 满足关系 lts s .tr
ue| and : 满足关系 lts s φ₁ → 满足关系 lts s φ₂ →满足关系
lts s (.and φ₁ φ₂)| or₁ : 满足关系 lts s φ₁ → 满足关系 lts s
(.or φ₁ φ₂)| 菱形模态 : lts.Tr s μ s' →满足关系 lts s' φ →满
足关系 lts s (.菱形模态 μ φ)| /- ...-/
```

请注意，非正式的介绍假定了一个固定的标号迁移系统，这在我们的形式化中对应于一个参数。当标号迁移系统在上下文中明确时，我们可以交替使用 `lts.Satisfiess` φ 或 $s \models \varphi$。

定义满足关系后，我们可以证明否定的正确性。

定理 neg_满足关系 : ¬满足关系 lts s φ.neg ↔ 满足关系 lts s φ

3.3 指称语义

HML 具有组合式指称语义，将每个命题映射到满足它的状态集合。

$$[\![\text{true}]\!] = S \quad \text{(the set of all states)} \qquad [\![\text{false}]\!] = \emptyset$$
$$[\![\varphi_1 \wedge \varphi_2]\!] = [\![\varphi_1]\!] \cap [\![\varphi_2]\!] \qquad [\![\varphi_1 \vee \varphi_2]\!] = [\![\varphi_1]\!] \cup [\![\varphi_2]\!]$$
$$[\![\langle\mu\rangle\varphi]\!] = \{s \mid \exists s'.\, s \xrightarrow{\mu} s' \wedge s' \in [\![\varphi]\!]\} \qquad [\![[\mu]\varphi]\!] = \{s \mid \forall s'.\, s \xrightarrow{\mu} s' \Rightarrow s' \in [\![\varphi]\!]\}$$

将其翻译为 Lean 十分直接，如下所示。

```
定义 指称（φ : 命题 标签）(lts : 标号迁移系统 状态 标签):
    设置 状态:= 匹配 φ 与| .真 =>Set.univ | .假命题 => ∅.且 φ₁ φ₂ => φ₁.
denotation lts ∩ φ₂.denotation lts| .或 φ₁ φ₂ => φ₁.denotation lts ∪ φ₂.
denotation lts| .菱形模态 μ φ => { s | ∃s', lts.Tr s μ s' ∧ s' ∈ φ.
denotation lts }| .方框模态 μ φ => { s | ∀ s', lts.Tr s μ s' →s' ∈ φ.
denotation lts}
```

3.4 理论与理论等价

满足关系引出了理论、理论等价（在以往的工作中有时称为“逻辑等价”）以及区分命题等概念。

定义 5（理论与理论等价）。状态 s的理论是它所满足的所有命题的集合：理论(s) $= \{\varphi \mid s \models \varphi\}$。如果理论$(s_1)$ $=$ 理论(s_2)，则称两个状态理论等价，记作理论等价(s_1, s_2)。

我们将其翻译为：

定义理论(lts : 标号迁移系统 状态 标签) (s : 状态) : 集合 (命题 标签) : =
{ φ |lts.Satisfies s φ }定义理论等价 (lts : 标号迁移系统 状态 标签) (s_1 s_2 : 状态) :命题 : =理论 lts s_1 = 理论lts s_2

在我们的开发中，一个有用的引理是：对于任意两个并非理论等价的状态，总存在一个区分命题。

引理 非_理论等价_满足 (h : ¬ 理论等价 lts s1 s2):∃ φ, (满足 lts s1 φ ∧ ¬满足 lts s2 φ)

否定有助于在此得到一个优雅的表述——若没有它，就必须是 ∃ φ, (满足 lts s1 φ ∧ ¬满足 lts s2 φ) ∨ (¬满足 lts s1 φ ∧ 满足 lts s2 φ)，从而使后续证明更为复杂。

4 元理论

我们转向关键结果，这些结果形式化了 HML 的元理论。

4.1 指称语义的正确性

归纳满足关系与指称语义一致。

定理 1（语义等价）。 $s \models \varphi$ 当且仅当 $s \in [\![\varphi]\!]$。

Lean 证明遵循文献中的相同思路——我们按照命题的结构进行归纳。（花括号表示隐式参数，Lean 会在 elaboration 过程中尝试从上下文中找到它。）

定理 满足_属于_指称 {lts : 标号迁移系统 状态 标签} : 满足关系 ltss φ ↔ s ∈ φ.指称 lts := 通过归纳法 φ 对 s 作一般化 <; >grind

由于我们的代码、Lean 库和 Mathlib 提供了充分的基础设施（注解），`grind` 策略可以自动处理所有情况：它能展开我们的定义，并处理集合上的所有必要操作。

4.2 双模拟不变性

HML 的一个关键性质是，互模拟等价的状态在逻辑上无法被区分。

定理 2（双模拟不变性）。如果 R 是一个互模拟且 $s_1 \, R \, s_2$，那么 $s_1 \models \varphi$ 蕴含 $s_2 \models \varphi$。

该定理在 Lean 中的表述如下。

引理 互模拟_满足 {hrb : lts.IsBisimulation r} (hr : r s1 s2) (φ : 命题 标签) (hs : Satisfies lts s1 φ) :Satisfies ltss2 φ

证明对 φ 使用归纳法。所有情形都可通过 `grind` 解决，只需稍作提示，即可在菱形模态情形下跟踪互模拟。

利用这个定理以及互模拟等价具有对称性这一事实，我们可以证明互模拟蕴含理论等价。

推论 1（互模拟蕴含理论等价）。若 R 是一个互模拟，且 $s_1\ R\ s_2$，那么理论等价(s_1, s_2)。

```
引理 互模拟_TheoryEq {lts : 标号迁移系统 状态 标签} {hrb : lts.IsBisimulation r}
(hr :r s1 s2) : TheoryEqlts s1 s2
```

4.3 互模拟等价的刻画（Hennessy–Milner 定理）

推论 1 的逆命题——即理论等价蕴含互模拟等价——仅对像有限标号迁移系统成立。这是因为，利用 HML 联结词，只能完整描述某个状态的像中的有限个行为。证明通过展示理论等价本身就是一种互模拟来完成。

引理 1. 在任何像有限标号迁移系统中，理论等价 是一种互模拟。

利用 CSLib 的 API，我们将该引理表述如下。

```
引理 theoryEq_isBisimulation (lts :LTS State Label)[image_
finite: ∀ s μ, Finite (lts.image s μ)] :
lts.IsBisimulation(TheoryEq lts)
```

证明这一点比证明其他结果需要更多创造性，但我们仍然可以依赖一个标准论证。假设状态 s_1 and s_2 在理论上等价，并考虑例如 $s_1 \xrightarrow{\mu} s_1'$ for some μ ands_1' 的情形。我们通过证明 s_2 无法匹配这一迁移会导致矛盾来继续论证。也就是说，我们（荒谬地）假设 μ-导出状态 ofs_2 中的每一个都能与 s_1' 区分。由标号迁移系统是像有限的这一事实，我们可以（非计算地）收集所有通过 μ-迁移从 s_2 到达的状态，为每个状态得到一个区分命题 ϕ_i，并将它们合成为一个命题$\langle\mu\rangle \bigwedge_{i=1}^{n} \phi_i$（其中 n 是 s_2 的像的基数）。随后我们证明，该命题可以区分 s_1 和 s_2，从而得到矛盾。

Hennessy–Milner 定理立即得出。

定理 3（Hennessy–Milner）。对于任意像有限标号迁移系统，理论等价 $=$ $\sim$。

Lean 翻译是直接的。

```
定理 理论等价_等于_互模拟等价（lts : 标号迁移系统 状态 标签）[像_有限:
 ∀ s μ, Finite(lts.image s μ)] : 理论等价 lts = 互模拟等价 lts
```

我们的所有定义和结果都可以直接应用于 CSLib 的其他部分，以及使用标号迁移系统 API 的下游项目。例如，我们可以证明某些自动机和并发进程满足 HML 命题，而这些系统也继承了我们的元理论结果。下一个代码片段展示了，在 CSLib 的 CCS 中，理论等价与互模拟等价相一致（这由 [10] 得出）。

示例 `[∀ p μ, Finite ((CCS.lts (defs := d)).image p μ)] : 理论等价 (CCS.lts (defs := d)) =互模拟等价 (CCS.lts (defs := d)) : =理论等价_等价_互模拟等价 _`

该示例适用于任意一组 CCS 递归过程定义 `d`，只要其满足像有限性条件（例如，使用守卫递归 [10]）。

5 相关工作与结论

HML 的各种变体与扩展已在不同的定理证明器中得到形式化 [7,4,11,6]。与本文不同，这些工作并不侧重于证明 HML 的一般元理论结果，例如 Hennessy–Milner 定理。

与我们最接近的工作在 Isabelle 中形式化了 HML 的一种扩展，用以处理一种新型迁移系统，称为名义转换系统，并包含该框架下 Hennessy–Milner 定理的证明 [9]。相比之下，我们的开发既忠实于原始理论，又简单得多，但仍足以在完全一般的情形下证明 HML 的元理论（适用于任意标号迁移系统）。此外，我们的工作与 CSLib 中的 API 相集成，使我们的结果能够直接应用于所有使用相同 API 的项目（例如，CSLib 中给出的自动机理论与 CCS，以及下游项目）。

其中的复杂性源于这样一个目标：对诸如 π-演算及其扩展等进程演算进行推理。在这些演算中，迁移产生的导出进程可能依赖于迁移标签引入的变量名（约束变量）。要有效地推理此类进程，就需要扩展 HML，使命题能够依赖于菱形与方框模态中所写迁移标签的约束变量。我们将这一扩展以及针对其他特性（如其他类型的依赖关系）的扩展留待未来工作。对于 CSLib，一个重要的设计问题是如何提供一个通用框架，以支持面向不同演算的 HML 各类扩展。有趣的未来工作还包括扩展至弱等价和不动点算子。

我们的开发工作是 CSLib 中首个模态逻辑形式化，并作为该库当前的 Hennessy–Milner 逻辑参考实现，为未来在库内开展扩展和应用提供可靠基础。

致谢。 我们感谢 Christopher Henson 对代码提出意见。Alexandre Rademaker 谨此感谢 Schmidt Sciences 提供的支持。本项目由欧盟（欧洲研究委员会，CHORDS，101124225）共同资助。本文所表达的观点和意见仅代表作者本人，并不一定反映欧盟或欧洲研究委员会的观点和意见。欧盟和资助机构均不对此承担责任。

参考文献

1. Aceto, L., Ingólfsdóttir, A.: 递归 Hennessy-Milner逻辑的测试。载于：Thomas, W.（编）《软件科学与计算结构基础》，

第二届国际会议，FoSSaCS’99，作为欧洲软件理论与实践联合会议 ETAPS’99 的一部分于荷兰阿姆斯特丹举行，1999年3月22日至28日，会议论文集。计算机科学讲义，第1578卷，第41–55页。Springer（1999）。https://doi.org/10.1007/3-540-49019-1_4, 10 1007 3 540 49019 1_4 https://doi.org/ 。 / - - - . Barrett, 2
C., Chaudhuri, S., Montesi, F., Grundy, J., Kohli, P., de Moura, L., Rade- maker, A., Yingchareonthawornchai, S.: CSLib：Lean计算机科学库（2026 04846 3 2602），https://arxiv.org/abs/ . . mathlib社区，T.: Lean 数学库。载于：第9届 ACM SIGPLAN 认证程序与证明国际会议论文集。第367–381页。CPP 2020，美国计算机协会，纽约，NY，美国（2020）。https://doi.org/10.1145/3372885.3373824, https://doi.org/10. 1145/3372885.3373824 4. Coupet-Grimal, S.: 归纳构造演算中线性时序逻辑的公理化。J. Log. Comput. **13**(6), 801–813 (2003). https://doi. org/10 801 5 1093 13 6 801, 10 1093 13 6 . /LOGCOM/ . . https://doi.org/ . /logcom/ . . . Hennessy, M., Milner, R.: 非确定性与并发的代数定律。ACM期刊 **32**(1), 137–161 (1985) 6. van Hulst, A.C., Reniers, M.A., Fokkink, W.J.: Hennessy-Milner逻辑的极大综合。ACM Trans. Embed. Comput. Syst.**14**(1), 10:1–10:21 (2015). https: //doi.org/10 2680540 7 1145 2680540, 10 1145 . / https://doi.org/ . / . Miculan, M.: 归纳构造演算中模态 μ-演算的形式化。Inf. Comput. **164**(1), 199–231 (2001). https://doi.org/ 10 2902 8 1006 2000 2902, 10 1006 2000 . /INCO. . https://doi.org/ . /inco. . . Milner, R.:《通信系统演算》，计算机科学讲义，第92卷。Springer（1980）。https://doi.org/10.1007/3-540-10235-3, https: //doi.org/10 3 9 1007 3 540 10235 . / - - - . Parrow, J., Borgström, J., Eriksson, L., Gutkovas, R., Weber, T.: 名义转换系统的模态逻辑。Log. Methods Comput. Sci. **17**(1) (2021), https: //lmcs.episciences.org/7137 10. Sangiorgi, D.:《互模拟与余归纳导论》。剑桥大学出版社，剑桥（2011）11. Tsai, M.H., Wang, B.Y.: 归纳构造演算中 CTL* 的形式化。载于：Okada, M., Satoh, I.（编）《计算机科学进展——ASIAN 2006：安全软件及相关问题》。第316–330页。Springer 海德堡，柏林，海德堡（2007）