用于数学软件的公平文件格式

Antony Della Vecchia[1][0009-0008-1179-9862],
Michael Joswig[1,2][0000-0002-4974-9659], and
Benjamin Lorenz[1][0000-0003-2648-718X]

[1] Technische Universität Berlin, Chair of Discrete Mathematics/Geometry, Berlin, Germany
[2] Max Planck Institute for Mathematics in the Sciences, Leipzig, Germany

摘要 我们描述了一种基于 JSON 的文件格式，用于在计算机代数中存储和分享结果而不丢失精度。根据实际可用性指导，一些关键特性包括处理设计时未知的数据结构的灵活性，一种清晰的方法来过渡到最新格式以及区分具有不同甚至矛盾语义数据的方式。这已在计算机代数系统奥斯卡 [5,20] 中实现，但我们还指出了如何在不同的环境中使用它。

我们讨论了总体考虑，重点是可理解性和长期存储。数据序列化的通用概念，如 Protocol Buffers[3] 或朱莉娅 的 [2] 序列化.jl ，不足以表示计算机代数的丰富语义。专门的软件系统确实允许存储和写入包含有限类型数学数据的文件，例如在优化中使用的 mps 文件格式，用于存储线性和整数规划。因此，这允许共享数据，例如，在数据库如 MIPLIB [16] 中。然而，像 mps 这样的格式不适用于更通用的数据。计算机代数系统的当前标准是使用笔记本存储整个计算过程，Jupyter 笔记本 [4] 是当前的标准。虽然这些笔记本非常方便，但它们并没有提供中间结果的适当序列化，这可能会使某些重新计算变得不理想或不可能。

20 世纪 90 年代末，开放数学 项目 [8] 开发了一个数学数据的通用框架。他们的工作受到了根本性的批评，例如 Fateman[10,11] 提出的批评。鉴于在开放数学 中提出的观点，在 [10] 中，我们选择了一个特定的系统，即用朱莉娅 编写的新的计算机代数系统奥斯卡 ，并依赖其语义，而没有试图一般性地形式化这些语义。我们将数据存储为注释树，这是综合序列化格式中的一个常见想法，例如 Protocol Buffers 和开放数学 。我们的格式扩展了当前的 JSON 文件格式，该格式为多面体软件包 [1]，它是原始 XML 版本 [14] 的翻

[3] https://protobuf.dev/
[4] https://jupyter.org

译。语法由可扩展的 JSON 架构固定，该架构在第 3 节中进行了说明。在第 4 节中，我们讨论了其他计算机代数系统的用户如何可能利用我们的格式。

我们的文件格式，作为数学研究数据计划 (MaRDI) [19] 的一部分开发，旨在最终扩展到超出奥斯卡 ，出于这个原因，我们使用了文件扩展名马迪。

1 通过示例的文件格式

我们的示例是一个在有限域上的二元多项式，即

$$2y^3z^4 + (a+3)z^2 + 5ay + 1 \ \in \ \mathrm{GF}(49)[y,z] \ , \tag{1}$$

其中 GF(49)，即包含 49 个元素的有限域，被构造成在素域 $\mathrm{GF}(7) \cong \mathbb{Z}/7\mathbb{Z}$ 上的一个二次代数扩张。更精确地说，作为 $(\mathbb{Z}/7\mathbb{Z})$-代数，GF(49) 与商 $(\mathbb{Z}/7\mathbb{Z})[x]/\langle x^2+1\rangle$ 同构，并且 x^2+1 是一个极小多项式。在后一个商代数中，我们选取一个生成元并称它为 a。由于域扩张的次数等于二，系数具有最大 a-次数为一。

```
{ "_ns": { "Oscar": ["https://github.com/oscar-system/Oscar.jl",
                     "1.0.0" ] },
  "_type": { "name": "MPolyRingElem",
             "params": "a7029443-b1d3-4708-a66d-f68eb6616fcf" },
  "data": [[["3", "4"], [["0", "2"]]],
           [["0", "2"], [["0", "3"], ["1", "1"]]],
           [["1", "0"], [["1", "5"]]],
           [["0", "0"], [["0", "1"]]]],
  "_refs": {
    "a7029443-b1d3-4708-a66d-f68eb6616fcf": { ... },
    "f2b7cb6b-535a-4a52-a0cc-75f8e93a6719": { ... },
    "23f25330-83f7-43a0-ac74-da6f2caa7eb8": {
      "_type": "FqField",
      "data": { "def_pol": {
        "_type": { "name": "PolyRingElem",
                   "params": "f2b7cb6b-535a-4a52-a0cc-75f8e93a6719" },
        "data": [["0", "1"], ["2", "1"]]
} } } } }
```

图 1. JSON 描述二元多项式 $2y^3z^4 + (a+3)z^2 + 5ay + 1$ 在多项式环 $\mathrm{GF}(49)[y,z]$ 中来自 (1)。我们隐藏了除一个描述定义多项式 x^2+1 的商域以外的所有参考文献。

我们的编码记录了整个构造的历史。因此，当我们存储那个多项式时，我们同时也存储了一元多项式环 $(\mathbb{Z}/7\mathbb{Z})[x]$、最小多项式 x^2+1 和商代数 $(\mathbb{Z}/7\mathbb{Z})[x]/\langle x^2+1\rangle$。这样，我们可以将代数表达式 (1) 与一棵反映其构造的注释树关联起来。这可以直接翻译成 JSON 代码，如图 1 所示。

我们区分了具有固定标准形式的基本类型。这包括标准的茉莉亚 类型，还包括诸如整数（ZZ 环元素）或有理数（QQ 域元素）这样的代数类型。更有趣的类型是参数化，多项式 (1) 的类型多项式环元素作为我们的示例。该类型将 (1) 标识为某个多元多项式环中的元素。它的系数环被引用为类型的参数多项式环元素，在那里它列在参数属性下。多元多项式环的基础环可以是任何环；在我们的示例中，这是由某个理想（由一个不可约多项式生成）商得的一元多项式环。这导致了一个递归描述，因为参数可以具有任何类型。因此，参数可能有自己的参数。所有参数类型及其参数等都存储在全局字典参考文献中，而参数属性通过通用唯一识别码（UUID）引用它们。我们在首次保存时根据 RFC 4122 生成版本四的 UUID，并且这些 UUID 在整个活动的奥斯卡 会话期间保持不变。例如，这使我们能够区分环的不同同构副本，它们可能在特定计算中扮演不同的角色。例如，当我们从两个多项式 $p \in \mathbb{Q}[a,b]$ 和 $q \in \mathbb{Q}[x,y]$ 开始，并且后来希望在环 $\mathbb{Q}[a,b,x,y]$ 中取它们的乘积时，这非常有用。它出现在日常的计算机代数程序中，其中“universe” $\mathbb{Q}[a,b,x,y]$ 不是预先知道的，而是几个计算步骤的结果。

有限域的例子说明，由于某些计算机代数构造（如数论中的赫森 lift）的增量性质，正规形式可能无法始终提供充分的描述；参见 [3] 和 [13, §15.4]。UUID 便于追踪这些演进计算。

因此，类型及其递归参数设置上下文，该上下文指定了序列化数据的语法。实际数据存储在同名属性中。在我们的示例中，数据子树的根有四个子节点，每个子节点对应多项式 (1) 的一个项。在 JSON 代码中，每个数据子树被写成嵌套列表的形式，并用方括号标记。

到目前为止的讨论涉及序列化的句法方面。语义完全隐式是关键的设计选择。在我们的示例中，语义由奥斯卡 版本 1.0.0 确定，该版本在命名空间属性_ns 中指定。命名空间构成了存储与奥斯卡 无关的数据的可能性的入口点。这是下面第 4 节的主题。

2 更多示例

为了展示我们概念所产生的可能性范围，我们挑选了两个性质截然不同的例子。

非一般类型的曲面在 $\mathbb{P}^4$ 中。已知每个光滑射影代数曲面都可以嵌入到射影 5-空间中，我们将其表示为 $\mathbb{P}^5$。根据 Ellingsrud 和 Peskine 的一个结果 [9]，只有有限个曲面族是非一般类型类型的，也就是说它们已经可以在 $\mathbb{P}^4$ 中嵌入。Decker 和 Schreyer 得出了 52 这个显式次数上限作为此类曲面的界限 [6]。在该文献中，作者还构造了 49 个非一般类型的曲面在 $\mathbb{P}^4$ 中，其度数不超过 15。该列表可通过存储在我方文件格式中的文件在 奥斯卡 中获得。

这里是一个这样的曲面（三次的，截线属为零），它被描述为一个理想在多项式环 $\mathrm{GF}(31991)[x, y, z, u, v]$ 中三个齐次生成元的消失点。下面的代码展示了一个使用奥斯卡 1.0.0 的交互式茉莉亚 会话。

```
julia> S = cubic_scroll()
Projective scheme
  over finite field of characteristic 31991
defined by ideal with 3 generators

julia> defining_ideal(S)
Ideal generated by
  31990*x*y + 19122*x*z + 4788*x*u + ... + 20742*u*v + 25408*v^2
  7471*x*y + 23772*x*z + 27471*x*u + ... + 30545*u*v + 9903*v^2
  x^2 + 3601*x*y + 7253*x*z + 7206*x*u + ... + 6535*u*v + 26586*v^2
```

将这些 49 个曲面的描述从文献转换为适合计算的对象需要一些时间，并且容易出错。因此，希望明确存储此类数据，无需任何计算或转换。

托里变换。以下代码构建了环面簇上的两个除子；请参见 [4]。A 托里环簇是一个代数簇，它由凸格多面体的法锥隐式描述，而一个托里坎除子是该多面体面的形式整系数线性组合，即其法锥的射线。我们的示例中的多面体是一个三角形，因此除子通过长度为三的整数向量给出，每个面向有一个条目。

```
@testset "ToricDivisor" begin
  pp = projective_space(NormalToricVariety, 2)
  td0 = toric_divisor(pp, [1,1,2])
  td1 = toric_divisor(pp, [1,1,3])
  vtd = [td0, td1]
  test_save_load_roundtrip(path, vtd) do loaded
    @test coefficients(td0) == coefficients(loaded[1])
```

```
    @test coefficients(td1) == coefficients(loaded[2])
    @test toric_variety(loaded[1]) == toric_variety(loaded[2])
  end
end
```

该测试保存和加载视图传输差异，这是一个由两个除子组成的向量，然后检查加载是否产生了相同的对象。此外，代码还使用 UUID 检查这两个除子的基础环面簇是否相同。

3 格式规范

JSON 模式 [22] 是一种声明性语言，用于描述 JSON 文件规范，类似于 RELAX NG 对于 XML 。我们的文件规范如图 2 所示。JSON 有四种类型，即字符串、数组、数字和对象（字典或哈希表）。第一次出现的类型属性描述了文件本身，其中它期望该文件类型为对象。属性和模式属性关键字用于描述对象的关键字和值的具体规格。只有在对象规范中匹配到的键（无论是精确匹配还是正则表达式匹配）对应的值才会被检查。要求关键字强制要求对象具有数组中列出的所有属性，这里我们强制要求类型属性存在。某些验证器可以处理常见的字符串格式，所以我们强制要求 参考文献 的键应具有 UUID 格式。其中一个 关键字用于指定列表中的一个规范是预期的。$ref 关键字使用路径或 URL 来引用在其他地方定义的规范，# 符号表示根目录。其他定义可以使用 $定义 s 部分进行描述。例如，我们对数据的定义接受多个选项，包括递归的对象和数组结构以及符合多面体软件包 模式的格式数据。

我们使用 UUID 而不是简单的索引，以便在整个会话中引用都有效。考虑以下场景，Alice 计算几个多项式，例如系数在某个固定的有限域中的多元多项式，如图 1 所示。然后她将三个这样的多项式的向量存储在一个文件中。进一步的计算则产生一个 3×3 矩阵，其系数位于相同的多项式环中。Alice 将该矩阵存储在另一个文件中。她将这两个文件发送给 Bob，后者希望继续进行这些计算，例如通过将矩阵与向量相乘。一般来说，有限域是作为素域上的字段扩展序列构建的。虽然对于任何给定的阶只有一个有限域，但有许多领域塔楼可以通向同一个结果。由于编码依赖于构造的细节，因此必须提供整个上下文。UUID 允许识别多个文件中的相同基本环。这对于数据库和大规模并行计算特别有用。

```
{ "$id": "https://oscar-system.org/schemas/mrdi.json",
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "type": "object",
  "required": ["_type"],
  "properties": { "_ns": { "type": "object" },
                  "_type": {
                    "oneOf": [
                      { "type": "string"},
                      { "type": "object", "properties": {
                        "name": {"type": "string"},
                        "params": {"$ref": "#/$defs/data"}
                      } } ] },
    "data": {"$ref": "#/$defs/data"},
    "_refs": {"type": "object", "patternProperties": {
      "^[0-9a-fA-F]{8}-([0-9a-fA-F]{4}-){3}[0-9a-fA-F]{12}$": {
        "$ref": "#"
      } } } },
  "$defs": { "data": {
    "oneOf": [
      { "type": "string"},
      { "type": "array", "items": { "$ref": "#/$defs/data"} },
      { "type": "object", "not": { "required": [ "_ns" ] },
        "patternProperties": {
          "^[a-zA-Z0-9_]*": {"$ref": "#/$defs/data"} } },
      { "$ref": "https://polymake.org/schemas/data.json"}
    ] } } }
```

图 2. 文件格式规范遵循 JSON Schema 规范 [22]。

4 超出奥斯卡

马迪文件格式旨在具有广泛的适用范围，而奥斯卡 主要作为概念验证。这里有一些尚未涵盖的方面。

命名空间。奥斯卡 是基于并扩展了内莫/赫克 [12] ，间隙 [18]，多面体软件包 和奇异的 [7]。因此，它涉及数论、群论和表示论、多面体几何与优化以及交换代数和代数几何中的精确计算。由于我们的文件格式将其语义委托给特定软件系统的特定版本，目前我们仅考虑“algebraic”数据。在 [10] 中，Fateman 指出“Sin[x]”在数学软件 [5] 和“sin(x)”在枫树 [6] 中表示非常不同的含

[5] https://www.wolfram.com/mathematica
[6] https://maplesoft.com/products/maple

义。这使得定义并利用涵盖此类数据的任何正式语义变得困难，超越了一个软件。在我们的文件格式中，可以通过为数学软件 Mathematica 和枫树 定义单独的命名空间来解决这个问题。

甚至将不同命名空间的数据放在同一个文件中也是有意义的。任何软件系统都可以自由地解释其能够理解的内容，并忽略其余部分。通过底层树结构，“the rest”可以引用任意子树。这样，马迪文件格式便成为一种灵活的容器格式，其精神类似于便携式文档格式（PDF）。[7] 例如，一个 PDF 文件可能包含音频数据，但并不是每个 PDF 查看器都具备播放声音的能力。

为了展示这在实践中如何工作，我们在图 3 中显示了一个简短的代码片段，该片段从一个平均直径文件中读取具有有理数系数的多元多项式到 SageMath [21]。那段代码是完全功能齐全且完整的，没有捷径或隐藏部分。

```
import json
from sage.all import PolynomialRing, QQ, prod

def load_oscar_polynomial(path):
  with open(path) as json_file:
    file_data = json.load(json_file)
    if ("Oscar" in file_data["_ns"] and
        file_data["_ns"]["Oscar"][1].startswith("1.0.")):
      t, d, refs = (file_data[k] for k in ["_type", "data", "_refs"])
      parent_ring_data = refs[t["params"]]["data"]
      base_ring = parent_ring_data["base_ring"]
      if base_ring["_type"] != "QQField":
        raise NotImplementedError("only rational coefficients supported")
      symbols = ",".join(parent_ring_data["symbols"])
      R, gens = QQ[symbols].objgens()
      p = R(0)
      for e, c in d:
        exps = [int(exponent) for exponent in e]
        coeff = QQ(c.replace("//", "/"))
        p += coeff * prod([g**i for g, i in zip(gens, exps)])
      return p
    else:
      raise RuntimeError("can only load OSCAR version 1.0 polynomials")
```

图 3. Python 代码用于从使用奥斯卡 1.0 编写的马迪文件中加载有理多项式，版本为 SageMath 10.2

[7] https://pdfa.org/resource/iso-32000-pdf/

查看 Python 代码也使我们能够解释如何避免 Fateman 对开放数学 [10,11] 的批评。如图 3 所示的代码明确地从奥斯卡 翻译到 SageMath 。这需要程序员了解这两个系统。在上世纪 90 年代，计算机代数系统之间的直接通信被认为是缺点之一，这也是推动开放数学 克服这一障碍的主要动机。像开放数学 这样的集中式通信概念所要付出的代价是需要正式指定语义。每个计算机代数系统都有丰富的隐含语义，这些语义通常非常难以明确表述。因此，开发一个能够同时管理几个计算机代数系统的形式化语义似乎至少与从头开始编写全新的计算机代数系统一样复杂。因此，我们的结论是坚持使用一个系统的隐含语义，例如奥斯卡 ，并在必要时进行显式翻译。这样 Fateman 的批评就不再适用。

对于有理多项式，从奥斯卡 翻译到另一个计算机代数系统的工作量相当小，如图 3 所示。根据数据类型的不同，其他转换可能需要更多的工作。命名空间为每个用户提供灵活性，使其可以选择自己的起点，即选择不同于奥斯卡 的计算机代数系统。

数据库。任何序列化都适合将相似的文件存储在某种系统化的文件夹层次结构中，模仿一个简单的数据库。我们的文件格式也不例外，第 1 节中的代数曲面在奥斯卡 中形成了一个示例。无 SQL 数据库 MongoDB 使用一种记录结构，这种结构本质上与 JSON 对象一致。这样，我们的序列化数据可以直接用于高效的大规模数据库的存储和检索。同一概念已经在多面体软件包的数据库项目多边形数据库 [17] 中被利用。请注意，MongoDB 需要 JSON 对象中的字符串采用 UTF-8 编码，因此我们也限制文件格式为 UTF-8。

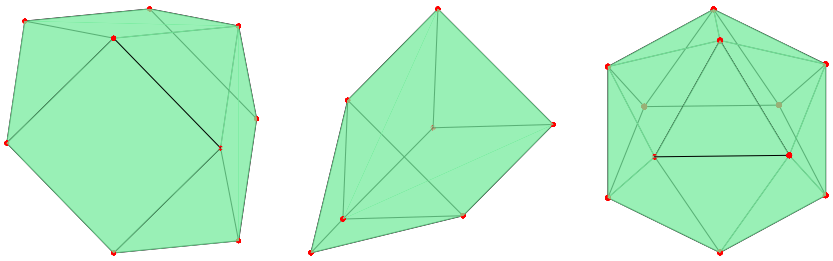

图 4. (a) 三角杯反角柱，(b) 延长的三角金字塔，(c) 旋延长五角锥

另一组有趣的马迪文件是来自 [15] 的 Johnson 立体。后者是三维凸多面体，其侧面为正多边形。Johnson 立体推广了 Archimedean 立体，总共有

92 种（在刚性运动和缩放下）并非 Archimedean 立体。数据集 [15] 包含作为这些 92 个多面体坐标的确切代数数以及近似浮点数。此外，数据附带一个朱莉娅 脚本，允许用户仅使用标准的茉莉亚 和 JSON 解析来读取 Johnson 固体的某些属性，独立于奥斯卡 。由于军衔文件格式基于 JSON ，因此有更简单的方法来访问该数据集的基本信息。例如，通过 jq，[8] 这个命令行 JSON 处理器。以下三个命令分别打印出三角穹顶、延长的三角金字塔和螺旋延长的五角金字塔的顶点数。这三种约翰逊多面体显示在图 4 中。

```
> jq '.data.float.VERTICES | length' j3
9
> jq '.data.float.VERTICES | length' j7
7
> jq '.data.float.VERTICES | length' j11
11
```

版本升级。奥斯卡 项目本身不应被视为整体的或完整的。相反，奥斯卡 被设计成能够持续进化，包括新的数据类型、编码和用例。其中一些变化表明需要修改数据的序列化方式，并且文件格式必须能够适应这些变化。为此，我们的数据附带版本号。升级脚本提供了从旧数据到当前标准的任意转换，而与 Protocol Buffers 的格式向前兼容不同。polymake 项目中已实施这种升级方案超过十年。在 2020 年版本 4.0 中，多面体软件包 替换了之前的 XML 基于的序列化为 JSON ，通过相同的机制。这表明此类升级是可行的。

元数据。当前文件格式的版本可以可选地附加元数据，其中包括数据名称和作者 ORCID 的条目。[9] 我们认为这足以作为文件格式的第一个版本，并且将根据 MaRDI 门户的要求进行更改。[10] MaRDI 门户旨在提供服务以提高数学研究数据的可发现性和可访问性。

5 结论与展望

我们设计的主要特点是不依赖于任何特定的编程语言。这一点本身就使其与朱莉娅 的序列化.jl 或 Python 的腌制品模块区分开来。将数学数据转换为 JSON 对象，这些对象只是字符串，总是意味着会有额外开销。对于长期存储空间效率通常不太相关，而其他特性更为重要。然而，一旦数据变得如

[8] https://jqlang.github.io/jq/
[9] https://orcid.org
[10] https://portal.mardi4nfdi.de/wiki/Portal

此之大以至于触及物理界限，例如硬盘的容量时，就必须考虑数据压缩了。通过命名空间，我们的方法允许压缩 `JSON` 对象的子树，并使用 `Base64` 二进制到文本编码来获得一个新的有效 `JSON` 对象。更详细的讨论超出了本文的范围。

致谢

我们感谢整个奥斯卡 开发团队实施和讨论代码；特别感谢 Claus Fieker、Tommy Hofmann 和 Max Horn。进一步地，我们感激 Lars Kastner 讨论了 FAIR 原则，感谢 Wolfram Decker 解释了在 $\mathbb{P}^4$ 中的代数曲面，并感谢 John Abbott、Ewgenij Gawrilow 和 Aaruni Kaushik 提供有用的反馈。

参考文献

1. Assarf, B., Gawrilow, E., Herr, K., Joswig, M., Lorenz, B., Paffenholz, A., Rehn, T.: Computing convex hulls and counting integer points with `polymake`. Math. Program. Comput. **9**(1), 1–38 (2017)
2. Bezanson, J., Edelman, A., Karpinski, S., Shah, V.B.: Julia: A fresh approach to numerical computing. SIAM Review **59**(1), 65–98 (2017)
3. Buchberger, B., Loos, R.: Algebraic simplification. In: Computer algebra, pp. 11–43. Springer, Vienna (1983)
4. Cox, D.A., Little, J.B., Schenck, H.K.: Toric varieties, Graduate Studies in Mathematics, vol. 124. American Mathematical Society, Providence, RI (2011)
5. Decker, W., Eder, C., Fieker, C., Horn, M., Joswig, M. (eds.): The Computer Algebra System `OSCAR`: Algorithms and Examples. Algorithms and Computation in Mathematics, Springer (2024)
6. Decker, W., Schreyer, F.O.: Non-general type surfaces in $\mathbf{P}^4$: some remarks on bounds and constructions. J. Symbolic Comput. **29**(4-5), 545–582 (2000), symbolic computation in algebra, analysis, and geometry (Berkeley, CA, 1998)
7. Decker, W., et al.: Singular — A computer algebra system for polynomial computations, version 4.3.2-p16 (2024), http://www.singular.uni-kl.de
8. Dewar, M.: `OpenMath`: An overview. SIGSAM Bull. **34**(2), 2 – 5 (6 2000)
9. Ellingsrud, G., Peskine, C.: Sur les surfaces lisses de $\mathbf{P}_4$. Invent. Math. **95**(1), 1–11 (1989)
10. Fateman, R.: A critique of `OpenMath` and thoughts on encoding mathematics. https://people.eecs.berkeley.edu/~fateman/papers/openmathcrit.pdf (1 2001)

11. Fateman, R.: More versatile scientific documents. https://people.eecs.berkeley.edu/~fateman/MVSD.html (2003)
12. Fieker, C., et al.: Nemo/Hecke: Computer algebra and number theory packages for the Julia programming language. In: Proceedings of the 2017 ACM on International Symposium on Symbolic and Algebraic Computation. pp. 157–164. ISSAC '17, ACM, New York, NY, USA (2017)
13. von zur Gathen, J., Gerhard, J.: Modern computer algebra. Cambridge University Press, New York (1999)
14. Gawrilow, E., Hampe, S., Joswig, M.: The `polymake` XML file format. In: Mathematical software – ICMS 2016. 5th international congress, Berlin, Germany, July 11–14, 2016. Proceedings, pp. 403–410. Berlin: Springer (2016)
15. Geiselmann, Z., et al.: Algebraically precise Johnson solids (Mar 2024). https://doi.org/10.5281/zenodo.10729583
16. Gleixner, A., et al.: `MIPLIB` 2017: Data-Driven Compilation of the 6th Mixed-Integer Programming Library. Mathematical Programming Computation (2021)
17. Paffenholz, A.: `polyDB`: a database for polytopes and related objects. In: Algorithmic and experimental methods in algebra, geometry, and number theory, pp. 533–547. Springer, Cham (2017)
18. The GAP Group: GAP – Groups, Algorithms, and Programming, Version 4.12.2 (2022), https://www.gap-system.org
19. The MaRDI consortium: MaRDI: Mathematical Research Data Initiative Proposal (May 2022). https://doi.org/10.5281/zenodo.6552436
20. The `OSCAR` Team: `OSCAR` – Open Source Computer Algebra Research system, Version 1.0.0 (2024), https://www.oscar-system.org
21. The `SageMath` Developers: `SageMath`, version 10.2 (Dec 2023). https://doi.org/10.5281/zenodo.10252400
22. Wright, A., Andrews, H., Hutton, B., Dennis, G.: Json schema: A media type for describing json documents. https://json-schema.org/draft/2020-12/json-schema-core.html (2020)