心於至善

基于回答集编程 视觉问答技术研究与实现 贾梁 东南大学

学校代码: 10286
分 类 号: 000
密 级: 公开
U D C: 000
学 号: 222171

工程硕士学位论文

基于回答集编程的视觉问答技术研究与实现

(学位论文形式: 应用研究)

研究生姓名: **贾梁**

导 师 姓 名: **张志政 副教授**

申请学位类别 **工程硕士** 学位授予单位 **东 南 大 学**

工程领域名称 **电子信息** 论文答辩日期 **2025 年 5 月 30 日**

研 究 方 向 **计算机技术** 学位授予日期 **2025 年 6 月 20 日**

答辩委员会主席 **翟玉庆** 评 阅 人 **倪庆剑**

张祥

2025 年 2 月 11 日

学校代码: 10286
分 类 号: 000
密 级: 公开
U D C: 000
学 号: 222171

工程硕士学位论文

基于回答集编程的
视觉问答技术研究与实现

(学位论文形式: 应用研究)

研究生姓名: **贾梁**

导 师 姓 名: **张志政 副教授**

申请学位类别 **工程硕士** 学位授予单位 **东南大学**

工程领域名称 **电子信息** 论文答辩日期 **2025 年 5 月 30 日**

研 究 方 向 **计算机技术** 学位授予日期 **2025 年 6 月 20 日**

答辩委员会主席 **翟玉庆** 评 阅 人 **倪庆剑 张祥**

2025 年 2 月 11 日

东南大学

工程硕士学位论文

基于回答集编程的
视觉问答技术研究与实现

专 业 名 称: 电 子 信 息

研究生姓名: 贾 梁

导 师 姓 名: 张志政 副教授

RESEARCH AND IMPLEMENTATION OF VISUAL QUESTION ANSWERING TECHNOLOGY BASED ON ANSWER SET PROGRAMMING

A Thesis submitted to

Southeast University

For the Professional Degree of Master of Engineering

BY

Jia Liang

Supervised by:

Associate Prof. Zhang Zhizheng

School of Computer Science and Engineering

Southeast University

2025/2/11

东南大学学位论文独创性声明

本人声明所呈交的学位论文是我个人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得东南大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

研究生签名：_____ 日 期：_____

东南大学学位论文使用授权声明

东南大学、中国科学技术信息研究所、国家图书馆有权保留本人所送交学位论文的复印件和电子文档，可以采用影印、缩印或其他复制手段保存论文。本人电子文档的内容和纸质论文的内容相一致。除在保密期内的保密论文外，允许论文被查阅和借阅，可以公布（包括刊登）论文的全部或部分内容。论文的公布（包括刊登）授权东南大学研究生院办理。

研究生签名：_____ 导师签名：_____ 日 期：_____

摘　要

灵犀一指是一种非常厉害的武功。

关键词： 多模态，回答集编程，视觉问答

Abstract

powerful fingers is a kind of powerful kung fu.

Keywords: kung fu, theory, fundamental kung fu, powerful fingers

目录

摘　要 . . . . . **I**

Abstract . . . . . **III**

插图目录 . . . . . **VII**

表格目录 . . . . . **IX**

算法目录 . . . . . **XI**

术语与符号约定 . . . . . **XIII**

第一章　绪论 **1**

1.1　研究背景 1

1.2　相关研究现状 2

1.2.1　传统 VQA 方法 2

1.2.2　深度学习驱动的 VQA 2

1.2.3　基于多模态预训练的 VQA 2

1.2.4　可解释性与推理能力增强的 VQA 2

1.3　研究目标与内容 2

1.4　研究方法与技术路线 3

1.5　论文结构 3

第二章　背景知识 **5**

2.1　回答集编程 5

2.1.1　语法 5

2.1.2　语义 6

2.1.3　求解器 9

2.2　GLIP 9

2.3　大语言模型 10

2.4　DSPy 10

2.5　本章小结 11

第三章　数据集 **13**

3.1　物体属性 13

3.2　环境表示 13

3.3　场景表示 14

3.4 图像生成 14
3.5 问题表示 15
3.6 问题生成 15
3.7 本章小结 16

第四章 实验设计 19
4.1 回答集编程 19
4.1.1 语法 19
4.1.2 语义 19
4.1.3 求解器 19
4.2 GLIP 19
4.3 大语言模型 19

第五章 问答系统的构建 21
5.1 回答集编程 21
5.1.1 语法 21
5.1.2 语义 21
5.1.3 求解器 21
5.2 GLIP 21
5.3 大语言模型 21

第六章 结论与展望 23
6.1 回答集编程 23
6.1.1 语法 23
6.1.2 语义 23
6.1.3 求解器 23
6.2 GLIP 23
6.3 大语言模型 23

参考文献 25

插图目录

2.1 GLIP 结构示意图 11

3.1 生成环境以及该环境中的完整场景的流水线 15
3.2 生成部分场景和问题，并进行标记的流程 16

表格目录

2.1 扩展后的系统对比 10
3.1 约束模板 17

算法目录

术语与符号约定

KF	kung fu
PF	powerful fingers

第一章 绪论

1.1 研究背景

随着计算机视觉（Computer Vision）和自然语言处理（Natural Language Processing）技术的迅猛发展，跨模态智能理解成为人工智能研究的重要方向之一。其中，视觉问答（Visual Question Answering, VQA）[goyal2017making] 任务因其广泛的应用前景和挑战性，受到了学术界和工业界的广泛关注。

视觉问答是一种多模态任务，它要求计算机能够基于给定的图像内容，理解并回答关于该图像的自然语言问题。例如，给定一幅包含动物的图片，系统需要能够回答“这只动物是什么颜色?”或“图片中有几只猫?”等问题。这一任务的核心在于多模态信息的深度融合，即如何在视觉特征和语言信息之间建立有效的联系。

VQA 技术的研究涉及多个领域，包括计算机视觉中的目标检测（Object Detection）、图像分类（Image Classification）、图像语义分割（Semantic Segmentation），以及自然语言处理中的文本理解（Text Understanding）、语义解析（Semantic Parsing）等。近年来，深度学习的快速发展极大地推动了 VQA 技术的进步，尤其是基于卷积神经网络（Convolutional Neural Networks, CNNs）和循环神经网络（Recurrent Neural Networks, RNNs）的方法，使得计算机能够更好地理解和生成答案。

视觉问答在多个实际应用场景中具有重要价值。例如，在辅助盲人阅读图像信息、自主机器人理解环境、医疗影像分析、教育和娱乐等方面，VQA 技术都可以提供智能化的交互方式，提高系统的可用性和便利性。然而，由于数据的不确定性、语言表达的多样性、以及多模态信息融合的复杂性，VQA 仍然面临诸多挑战，例如开放域问题的泛化能力、推理能力的提升、以及对长尾问题的有效应对等。

为了解决这些挑战，研究人员提出了多种 VQA 模型架构，包括基于注意力机制（Attention Mechanism）[Bahdanau2015Attention]、图神经网络（Graph Neural Networks, GNNs）、预训练视觉-语言模型（Pretrained Vision-Language Models）等方法，以提高 VQA 系统的理解能力和回答质量。

大语言模型（Large Language Model）

回答集编程（Answer Set Programming，ASP）是一种声明式编程范式，可用于解决复杂的人工智能问题。ASP 起源于对逻辑编程、非单调推理和知识表示的研究。ASP 因其具有表达性声明性的语言和以 Clingo 为代表的一些高效实现而流行起来。ASP 已经在学术界和工业界得到广泛应用，并被证明在人工智能的几个知识密集型应用中能够有效解决问题，如调度、产品配置、机器人、劳动力管理和决策支持等。ASP 能够以简洁和直观的方式来表示知识，允许用户相对容易地去表示复杂问题，而且 ASP 具有分单调推理的特性，允许不完整信息的表示和默认推理。另外，ASP 对知识的表达能力，使得其支持集成各种类型的知识，包括规则、约束和偏好，有助于灵活解决问题。

1.2 相关研究现状

视觉问答（Visual Question Answering, VQA）作为一种多模态任务，近年来受到了广泛关注。研究者们提出了多种方法来提升 VQA 模型的性能，包括基于深度学习的端到端模型、注意力机制、图神经网络以及预训练的视觉-语言模型等。

1.2.1 传统 VQA 方法

早期的 VQA 研究主要依赖于特定的特征提取和简单的模型架构。例如，Antol 等人（2015）提出了最早的大规模 VQA 数据集 VQA v1，并基于长短时记忆网络（LSTM）与卷积神经网络（CNN）构建了一个简单的 VQA 基线模型[Antol2015VQA]。然而，这类方法在处理复杂推理任务时存在一定局限。

1.2.2 深度学习驱动的 VQA

近年来，随着深度学习的快速发展，基于神经网络的方法成为 VQA 研究的主流。Anderson 等人（2018）提出了基于注意力机制的 Bottom-Up and Top-Down（BUTD）模型，该方法在 VQA 任务上取得了显著提升[Anderson2018BUTD]。此外，Tan 和 Bansal（2019）探索了视觉和语言的联合预训练方法，提出了 LXMERT，一个专门为多模态任务设计的 Transformer 模型[Tan2019LXMERT]。

1.2.3 基于多模态预训练的 VQA

近年来，预训练模型在 VQA 任务中取得了突破性进展。例如，Radford 等人（2021）提出了 CLIP（Contrastive Language-Image Pretraining），通过大规模图像-文本数据进行对比学习，使得模型在零样本 VQA 任务上表现优异[Radford2021CLIP]。Li 等人（2020）提出了 UNITER，一种统一的视觉-语言表示学习框架，能够在多个 VQA 数据集上实现最先进的性能[Li2020UNITER]。

1.2.4 可解释性与推理能力增强的 VQA

虽然深度学习驱动的 VQA 模型在性能上取得了显著进步，但它们的可解释性和推理能力仍然是一个重要的研究问题。Gokhale 等人（2020）提出了一种基于因果推理的 VQA 方法，以减少数据偏差的影响[Gokhale2020CausalVQA]。此外，研究者们探索了基于知识图谱（Knowledge Graph）和符号推理的方法，以增强 VQA 的推理能力。例如，Hudson 和 Manning（2019）提出的 MAC 网络引入了可解释的记忆组件，以模拟逐步推理[Hudson2019MAC]。

1.3 研究目标与内容

本课题的主要研究目标是，实现一个视觉问答系统。具体研究内容如下：

(1) 神经-符号方法的设计。
(2) 设计并开发视觉问答系统。

1.4　研究方法与技术路线

本文针对以上研究目标和研究内容，综合多种方法进行研究。本文的研究主要涉及三个重点内容：

(1)

1.5　论文结构

本文共分为六个章节，各章节的主要内容具体如下：

第一章为绪论，总体介绍本文的研究背景及意义、相关研究现状与不足、研究目标与研究内容、研究方法与技术路线及本文的结构安排。

第二章为背景知识，对本文涉及到的主要技术进行介绍，具体包括 ASP 程序语法及 ASP 求解器、GLIP 以及 Dspy。

第三章为数据集构建。

第四章为基于回答集编程的视觉问答模型实现方案。

第五章为实验及结果分析。

第六章中对本文工作加以总结，并展望未来工作。

第二章　背景知识

2.1　回答集编程

2.1.1　语法

项（terms）是 ASP 程序中最基本的元素，项可以是常量、变量或者函数。常量以符号常量或者数字表示（如 4、a）。变量以字符串来表示，要求首字母必须大写（如 Dog、Person）。函数由函数符号与若干参数构成，例如 $f(t_1,...,t_n),(n \geq 0)$。各参数 $t_i,(i=1,2,...,n)$ 也是项。若项中不存在函数符号与变量，则称该项为实例化项，否则称该项为非实例化项。

原子（atom）由谓词和项共同组成，例如 $q(t_1,...,t_n),n \geq 0$。其中，q 为 n 元谓词的标识符，$t_1,...,t_n$ 为项，n 为 0 时，谓词之后的括号可以省略。原子用于表示不同项之间的关系，例如：$dog(animal)$、$family(jensen,lucy)$、$tomorrow\_sunny$，分别表示狗是动物、jensen 和 lucy 是家人以及明天晴天。如果原子中的每一项都是实例化的，则该原子是实例化原子，否则，该原子是非实例化的原子。例如，$dog(animal)$ 是实例化原子，$family(jensen,lucy)$ 是非实例化原子。原子 $q(t_1,...,t_n)$ 的强否定形式为 $\neg q(t_1,...,t_)n$，其中符号 $\neg$ 是经典逻辑中的否定，$\neg q(t_1,...,t_n)$ 表示各项 $t_i(1 \leq i \leq n)$。不符合 q 所描述的关系，$\neg\neg a = a$，与经典逻辑中的排中律相同。文字（literal）可以是原子 $q(t_1,...,t_n)$，也可以是原子的强否定形式 $\neg q(t_1,...,t_n)$。若文字对应的原子是实例化的，则称该文字是实例化的。

定义 2.1.1 (规则). 规则 r 是具有如下形式的式子：

$$l_0\, or ...\, or\, l_m \leftarrow l_{m+1},...,l_n\, not\, l_{n+1},...,\ not\, l_p. \tag{2.1}$$

其中 $p \geq n \geq m \geq 0, l_i$ 代表文字，not 为新的逻辑连接符，通常称作缺省否定符（default negation）或者失败即否定（Negation As Failure, NAF），又称为若否定，$not\, l_i$ 被读作“没有理由相信 l_i 为真”，但是这并不表明 l_i 为假。对 ASP 程序而言，$notnota$ 不与 a 等价。规则 r 读作“如果相信 $l_{m+1},...,l_n$ 为真，并且没有理由相信 $l_{m+1},...,l_p$ 为真，则相信 $l_0\, or\, ...\, l_m$ 为真”。将 $notl_i$ 称为缺省文字（default literal），文字与缺省文字共同组成拓展文字（extended literal）。

规则 r 左侧的文字称为头部，可以表示为 $head(r)=\{l_0,...,l_m\}$。规则 r 右侧的文字称为体部，可以表示为 $body(r)=\{l_{m+1},...l_n,not\, l_{n+1},...,not\, l_p\}$，规则的体部可以分为正体部与负体部，正体部表示为 $body^+(r)=\{l_{m+1},...,l_n\}$，负体部表示为 $body^-(r)=\{l_{n+1},...,l_p\}$ 。因此，规则 r 可以表示为：

$$head(r)\ \leftarrow\ body^+(r),\ notbody^-(r) \tag{2.2}$$

一个 ASP 程序是有限条(2.2)所示的规则组成的集合。根据规则各部分所满足的相关条件，可以分别定义事实、约束、正规规则、缺省规则和严格规则，如定义(2.1.2)-定义(2.1.6)所示。

定义 2.1.2 (事实). 当规则 r 满足 $head(r) \neq \emptyset, body(r)=\emptyset$ 时，被称为事实（*fact*）。

定义 2.1.3 (约束). 当规则 r 满足 $head(r)=\emptyset, body(r) \neq \emptyset$ 时，被称为约束（*constraint*）。

定义 2.1.4 (正则规则). 当规则 r 满足 $|head(r)| = 1$ 时，被称为正则规则（*normal rule*）。

定义 2.1.5 (缺省规则). 当规则 r 满足 $body(r)^- \neq \emptyset$ 时，被称为缺省规则（*default rule*）。

定义 2.1.6 (严格规则). 当规则 r 满足 $body(r)^- = \emptyset$ 时，被称为严格规则（*definite rule*）。

根据上述各类规则的定义，本文进一步定义简单逻辑程序、缺省逻辑程序和正规逻辑程序如下。

定义 2.1.7 (简单逻辑程序). 若程序 P 由有限条严格规则组成，则 P 被称为简单逻辑程序。

定义 2.1.8 (拓展逻辑程序). 若程序 P 由有限条缺省规则组成，则 P 被称为拓展逻辑程序。

定义 2.1.9 (正则逻辑程序). 若程序 P 除约束外，由有限条正则规则组成，则 P 被称为正则逻辑程序（*Normal Logic Program, NLP*）。

若不做特别说明，本文考虑的程序均为不含约束的正规逻辑程序，即程序中的每条规则头部有且仅有一个文字。

2.1.2 语义

文字的可满足性语义解释和程序的模型

作为基础理论，首先给出 Herbrand 域、Herbrand 基的定义，分别如定义 2.10 和 2.11 所示，进一步定义规则与程序的实例化，最终在实例化程序下讨论文字的可满足性语义解释和可满足性。

定义 2.1.10 (Herbrand 基). P 是一个 *ASP* 程序，将程序 P 中出现的常量和函数形成的所有不含变量的项的集合称为 *Herbrand* 域，使用 $\mathcal{U}_P$ 表示。

定义 2.1.11 (Herbrand 域). P 是一个 *ASP* 程序，出现在程序 P 的规则中的谓词符号与 $\mathcal{U}_P$ 中的项组成的所有可能的不含变量的原子集合称为程序 P 的 *Herbrand* 基，使用 $\mathcal{A}_P$ 表示，为了保持简洁，通常简写为 $\mathcal{A}$。

定义 2.1.12 (规则的实例化). 给定 P 中的一个规则 r，使用 $ground(r)$ 表示通过使用 $\mathcal{A}_P$ 中的常量对应地替换 r 中变量而获得的规则集合，这一映射被称为规则的实例化。

定义 2.1.13 (程序的实例化). 程序 P 由一系列规则的集合组成，程序 P 的实例化 $ground(P)$ 表示为式(2.3):

$$ground(P) = \bigcup_{r \in P} ground(r) \tag{2.3}$$

定义 2.1.14 (文字的可满足性语义解释). 定义文字的可满足性语义解释 $I = \langle I^+, I^- \rangle$, 其中 $I^+ \cup I^- \subseteq \mathcal{A}_P$ （P 是实例化程序），$I^+ \cap I^- = \emptyset$。I^+ 表示为已知为真的文字集合，I^- 表示已知为假的文字集合。当 $I^+ \cap I^- = \mathcal{A}_P$ 时，I 被称作完全可满足性语义解释（*complete interpretation*）。若 I 不是完全可满足性语义解释，则 I 中存在真值未确定（*undefined*）的文字。

定义 2.1.15 (程序的模型). 假设 P 是一个实例化的回答集程序，$I = \langle I^+, I^- \rangle$ 是一个可满足性语义解释。若文字 $l \in I^+$，则称 I 满足 l，记作 $I \models l$；对缺省文字 $notl$，若 $l \in I^-$，则称 I 满足 $notl$，记作 $I \models notl$。对于文字集合 S，若 $\forall l \in S, I \models l$, 则称 I 满足集合 S，记作 $I \models S$，当一个规则 r 满足 $I \not\models body(r)$ 或 $I \models head(r)$ 时，I 满足规则 r。当 I 满足程序 P 的所有规则时，称 I 是 P 的一个模型。

回答集语义

定义 2.1.16 (一致性文字集合). 给定一个文字集合 S，若 S 中不同时包含 l 和 $\neg l$，则集合 S 是一致性文字集合，其中，$l \in S$ 是集合 S 中的任意文字。

定义 2.1.17 (可满足性，Satisfiability). 给定一致性文字集合 S，文字 lit 与规则 r，若 $lit \in S$，则文字 lit 满足集合 S。若 $head(r) \cap S \neq \emptyset$，则 S 满足头部。若 $body^+(r) \subseteq S, S \cap body^-(r) = \emptyset$，则集合 S 满足体部。若 S 满足规则 r 的头部或者 S 不满足规则 r 的体部，则 S 满足规则 r，记作 $S \models r$。给定一个 *ASP* 程序 P，若 $\forall r \in P, S \models r$，则 S 满足程序 P。

定义 2.1.18 (规则的适用与阻塞). 给定一致性文字集合 S 与规则 r，S 满足 $body(r)$，则称规则 r 在集合 S 下是适用的（*applicable*），否则称规则 r 在集合 S 下被阻塞（*block*）。

例 2.1.1. 给定 *ASP* 程序 P_3，该程序包含两条规则：

$$r_1 : p \leftarrow q, s.$$
$$r_2 : s \leftarrow t.$$

下面通过分析说明集合 $S = \{p, q\}$ 对程序 P 的可满足性。

(1) 因为 $p \in S$，所以文字 p 与文字 q 均满足 S；
(2) 因为 $S \cap head(r_1) = \{p\} \neq \emptyset$，所以 S 满足 $head(r_1), S \models r_1$；
(3) 因为 $body^+(r_2) \subsetneq S$，所以 S 不满足 $body(r_2)$，所以 $S \models r_2$；
(4) 因为 S 满足程序 P 中的每一条规则，所以 S 满足程序 P。

定义 2.1.19 (Gelfond-Lifshitz 规则（GL 规约）). 假设程序 P 是给定的实例化程序，S 是一致性文字集合，l 是文字，$S \subseteq Lit(P), l \in Lit(P)$, P 关于 S 的 GL 规约结果 P^S 定义为式(2.4)：

$$P^S = \{head(r) \leftarrow body^+(r) \mid r \in P, body^-(r) \cap S = \emptyset\} \tag{2.4}$$

定义 2.1.20 (简单逻辑程序回答集[<empty citation>]). 假设 P 是简单逻辑程序，程序 P 的回答集是 $S \subseteq Lit(P)$ 满足：

(1) 对于程序 P 中的每一条规则 $l_0 \leftarrow l_1, ..., l_m$，如果 $l_1, ..., l_m \in S$，那么 $l_0 \in S$。
(2) $S \subseteq Lit(P)$ 且不存在 S 的任何子集也满足程序 P 的每一个规则，即 S 是满足程序 P 的最小集合；
(3) 如果 S 包含互补的文字，那么 $S = Lit(P)$；

定义 2.1.21 (拓展逻辑程序回答集). 假设 P 是包含缺省否定的扩展逻辑程序，S 是实例化文字集合，如果 S 是 P^S 的回答集，则 S 是 P 的回答集。如果程序 P 没有回答集或者只有一个包含互补文字的回答集 $Lit(P)$，那么程序 P 是不一致程序，否则，程序 P 是一致性程序。

可以看出，使用 *GL* 规约，可以将扩展逻辑程序程序转换为简单逻辑程序，从而得到扩展逻辑程序的回答集。下面通过一个例子说明。

例 2.1.2 (GL 规约与问答集). 给定一个实例化的 *ASP* 程序 P_4：

$$r_1 : p(a) \leftarrow notp(b)$$
$$r_2 : p(b) \leftarrow notp(a)$$
$$r_3 :\leftarrow p(b).$$

程序 P_4 可能的回答集有四个：$S_1 = \emptyset, S_2 = \{p(a)\}, S_3 = \{p(b)\}, S_4 = \{p(a), p(b)\}$, 根据回答集的定义依次验证：

(1) $P^{S_1} = \{p(a) \leftarrow .\} \cup \{p(b) \leftarrow .\} \cup \{\leftarrow p(b).\}$，事实 $(p(b) \leftarrow .)$ 与约束 $(\leftarrow p(b).)$ 同时在 P^{S_1} 中出现，因此 S_1 不是 P^{S_1} 的回答集，所以 S_1 不是 P 的回答集。

(2) $P^{S_2} = \{p(a) \leftarrow .\} \cup \{\leftarrow p(b).\}$，而 $S_2 \models ((p(a) \leftarrow .))$ 且 $S_2 \models (\leftarrow p(b).)$，因此 $S_2 \models P^{S_2}$，且不存在 $S_2' \subset S_2, S_2' \models P^{S_2}$，所以 S_2 是 P^{S_2} 的回答集；

(3) $P^{S_3} = \{p(b) \leftarrow .\} \cup \{\leftarrow p(b).\}$，事实 $(p(b) \leftarrow .)$ 与约束 $(\leftarrow p(b).)$ 同时在 P^{S_3} 中出现，因此 S_3 不是 P^{S_3} 的回答集，所以 S_3 不是 P 的回答集；

(4) $P^{S_4} = \{\leftarrow p(b).\}$，可知 P^{S_4} 的回答集为 $\emptyset$，因此 S_4 不是 P^{S_4} 的回答集，所以 S_4 不是 P 的回答集。

well-founded 语义

上世纪 90 年代，良基语义（well-founded semantics）的概念由 Van Gelder A 提出，对于任意的回答集程序，都存在对应的良基模型（well-founded models），并且在多项式时间内可以计算出这个模型[van1991well]。该语义模型是本文对 NAF 文字进行解释和对不一致程序原因进行分类的重要基础，下面介绍这一模型的定义及计算方法。

定义 2.1.22 (立即结论). P 是一个 *ASP* 程序，$S \subseteq \mathcal{A}_P \square V \subseteq \mathcal{A}_P$，则集合 S 关于 P 和 V 的立即结论（*immediate consequence*）记作 $\mathcal{T}_{P,V}(S)$，定义如式(2.5)所示：

$$T_{P,V}(S) = \{a | \exists r \in P, head(r) = a, body^+(r) \subseteq S, body^-(r) \cap V = \emptyset\} \tag{2.5}$$

考察上式可以发现，当集合 V 确定时，关于集合 S 的函数 $T_{P,V}(S)$ 是单调的。因此，当集合 V 固定时，该函数存在最小不动点（Least Fixed Point, LFP），使用 $lfp(.)$ 表示。

定义 2.1.23 (良基模型). P 是一个 *ASP* 程序，P^+ 是程序 P 中的全部严格规则的集合，序列 $(K_i, U_i)_{i \geq 0}$ 定义如下：

$$K_0 = lfp(T_{P^+}) \qquad K_i = lfp(T_P, U_{i-1})$$
$$U_0 = lfp(T_{P,K_0}) \qquad U_i = lfp(T_{P,K_i})$$

若 $\langle K_j, U_j\rangle = \langle K_{j+1}, U_{j+1}\rangle$，且不存在 $0 \leq k < j$，使得 $\langle K_k, U_k = \langle K_{k+1}, U_{k+1}\rangle$（j,k 是正整数），则 P 的良基模型 $WF_P = \langle W^+, W^-\rangle$，满足 $W^+ = K_j$，$W^- = \mathcal{A}\backslash U_j$

例 2.1.3 (良基模型). 程序 P_5 包含以下四条规则：

$$r_1 : q \leftarrow, notp. \qquad r_2 : p \leftarrow, notq.$$
$$r_3 : a \leftarrow b. \qquad r_4 : b \leftarrow .$$

首先得到 $P_5^+ = \{a \leftarrow b\} \cup \{b \leftarrow .\}$，而 $lfp(T_{P_5^+}) = a, b$，因此 $K_0 = a, b$；进一步计算得到 $U_0 = lfp(T_{P_5^+, K_0}) = \{a, b, p, q\}$，$K_1 = lfp(T_{P_5^+, U_0}) = \{a, b\} = K_0$，$U_1 = lfp(T_{P_5^+, K_1}) = \{a, b, p, q\} = U_0$。因此 $\langle K_0, U_0\rangle = \langle K_1, U_1\rangle$ 且不存在 $0 \leq k < 1$，$\langle K_k, U_k\rangle = \langle K_{k+1}, U_{k+1}\rangle$。因此程序 P_5^+ 的良基模型为 $WF_{P_5^+} = \langle \{a, b\}, \{\}\rangle$，$p$ 和 q 在良基模型中属于未定义（$undefined$）。

2.1.3　求解器

求解器是实现 ASP 语义推理的核心工具，其工作原理是：通过搜索逻辑程序的最小模型（即回答集）来解决复杂的组合优化问题。

求解器的工作流程可以划分为两个阶段：基础化（Grounding）和求解（Solving）。基础化是将一阶逻辑程序实例化为命题逻辑形式，而求解则是基于 CDCL 算法搜索满足约束的模型。

截至目前，已经有众多 ASP 求解器。学术界和工业界所用的主流的求解器，可分为两类：单阶段求解器（如 Clasp、DLV）、多阶段求解器（如 Clingo）。两类求解器的主要区别在于，单阶段求解器独立地进行基础化与求解，而多阶段求解器则是集成了基础化与求解的交互式系统。

表2.1对目前的主流求解器的适用场景及核心优势进行了对比。因为 Clingo 求解器具有免费开源、使用简单、运行高效的特点，本文在求解 ASP 程序时，使用 Clingo 进行相关实验。

2.2　GLIP

GLIP（Grounded Language-Image Pretraining）是一种融合视觉与语言的预训练模型，旨在提高模型在目标检测、图像理解以及跨模态任务上的能力。GLIP 通过对大规模的图像-文本对数据进行联合训练，使得模型能够理解自然语言查询，并在图像中定位相应的目标，从而实现零样本（zero-shot）或少样本（few-shot）的目标检测任务。其架构如图2.1所示。

GLIP 采用了大规模的图像-文本数据进行预训练，使用基于 Transformer 的架构，并结合视觉和语言的特征进行跨模态学习。其训练过程主要包括：（1）文本引导的目标检测：输入不仅包含图像，还包含文本描述，使得模型能够基于自然语言查询来识别目标。（2）跨模态对齐：通过对比学习和注意力机制，使得视觉特征与语言特征在共享的表示空间中进行对齐。（3）自监督学习：利用大规模无标注数据进行自监督学习，提高模型的鲁棒性和泛化能力。

相比以往的目标检测技术，GLIP 的突出优势在于：（1）零样本检测能力：无需重新训练，即可识别新的类别。（2）泛化性强：由于使用了开放词汇的文本描述，GLIP 在实际应用中更加灵活。（3）跨领域适应性：可以应用于医学影像分析、自动驾驶、机器人视觉等多个领域。

求解器	核心技术	适用场景	核心优势
Clasp	多线程 CDCL 情性子句学习	大规模组合优化 工业级应用	支持并行搜索 性能竞赛冠军
Clingo	增量式基础化 多阶段编程	动态规则生成 交互式推理	集成 Python API 支持在线更新程序
DLV	数据库优化 存在线量化推理	知识表示型问题 语义 Web	高效处理复杂规则 支持非确定域
WASP	分层弱约束优化 冲突驱动剪枝	偏好推理 多目标优化	加权约束高效处理
Smodels	稳定模型算法 部分求值	教学与研究 基础模型验证	算法透明易扩展
LPARSE	基础化预处理	规则实例优化	与 Smodels 配合使用
IDP	基于模型的推理 扩展一阶逻辑	复杂知识库管理	支持类型系统 高阶推理
Alpha	并行求解引擎	超大规模问题	GPU 加速搜索

表 2.1: 扩展后的系统对比

目前，GLIP 在多个计算机视觉任务中表现优异，典型的应用包括：开放词汇目标检测（Open Vocabulary Object Detection, OVOD）、跨模态信息检索、复杂视觉问答（Visual Question Answering, VQA）、自动标注和数据增强。

2.3 大语言模型

2.4 DSPy

DSPy（Declarative Self-improving Language Programs）是由斯坦福大学开发的一种编程框架，旨在通过声明式的方式优化大语言模型（LLMs）的推理能力。DSPy 提供了一种高效的途径，使开发者能够构建自我改进的自然语言处理（NLP）系统，而无需手动调整复杂的超参数或微调大模型。

DSPy 的核心思想是将模型的优化过程抽象为声明式编程，使得用户能够专注于高层任务，而底层优化（如提示工程、少样本学习等）由系统自动完成。其主要特性包括：（1）模块化设计：开发者可以定义不同的语言任务模块，DSPy 会自动优化这些模块的执行方式。（2）自适应优化：系统会根据反馈数据不断调整推理策略，提高模型的准确性和鲁棒性。（3）多

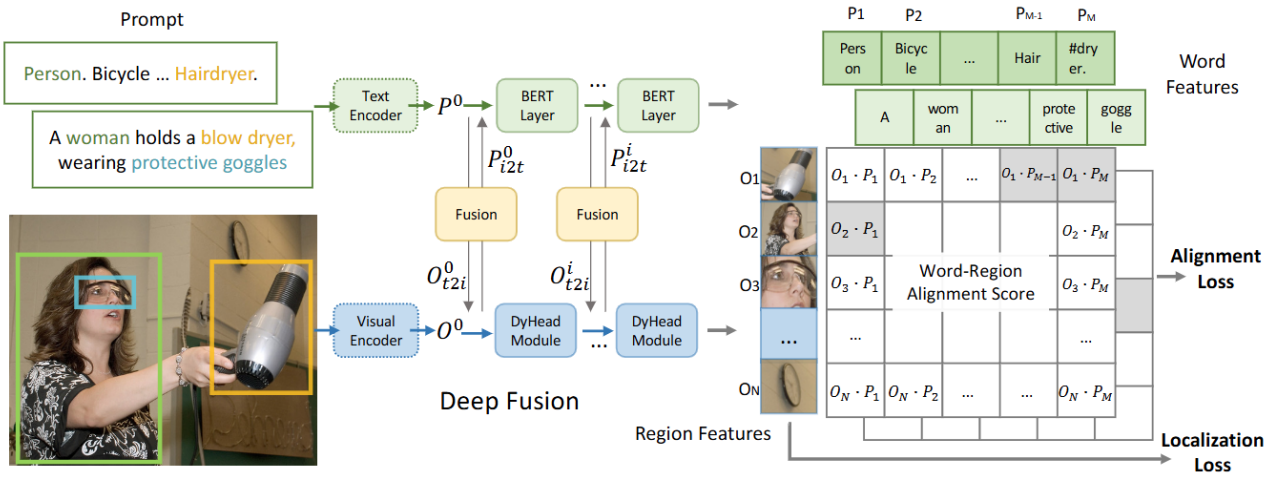


图 2.1: GLIP 结构示意图

模态支持：除了纯文本处理，DSPy 还可以用于多模态任务，如结合视觉和语言进行推理。

DSPy 采用了多种技术来提高大语言模型的推理效率和准确性，包括：（1）自动提示调整（Auto-Prompting）：根据任务需求，动态优化提示词，提高模型的响应质量。（2）少样本学习（Few-shot Learning）：利用少量示例进行任务适配，减少对大规模标注数据的依赖。（3）强化学习（Reinforcement Learning）：通过用户反馈不断优化推理过程，使模型的输出更加符合期望。

使用 DSPy 构建大语言模型应用，具有如下优势：（1）声明式编程范式：减少了对低层次参数调整的需求，使开发者可以更专注于任务逻辑。（2）高效优化：通过自动调整提示词和推理策略，提高大模型的推理能力。（3）可扩展性强：适用于各种 NLP 任务，包括文本摘要、信息抽取、代码生成等。

2.5　本章小结

第三章　数据集

本章详细描述基于 CLEVR 数据集，构造本文所用的 CLEVR-ASP 数据集的过程，以及数据集的设计思路、ASP 约束编码规则。

3.1　物体属性

CLEVR-ASP 数据集的图像中的每个物体，均有形状、尺寸、材质、颜色四种属性。每种属性的可能取值如下：

(1) 形状：圆锥体、球体。
(2) 尺寸：小、中、大。
(3) 材质：橡胶、金属。
(4) 颜色：红色、蓝色、绿色、黄色、灰色、棕色。

除了以上四种属性之外，由于图像中的场景被划分成 4 个区域，故 CLEVR-ASP 数据集也将图像中的物体按照其所在区域进行划分，分别标记为 0、1、2、3。每个物体都恰好位于某一个区域之中。将图像中的场景划分为多个区域，可以在多个级别上分别指定约束。

1. 区域约束：某个区域内的所有物体必须满足特定属性。例如，所有在区域 0 的物体必须是立方体或圆柱体。
2. 夸区域约束：定义了多个区域之间的关系。例如，区域 1 和区域 2 内同一个颜色的物体数量之和不得超过 2。
3. 全局约束：其中包含了一组影响整个场景的规则。例如，场景中至少有 2 个立方体。

以上所有约束，使用 ASP 来进行表示。例如：

```
:- object(X), at(X, 0), not hasProperty(X, shape, cube), not hasProperty
   ↪(X, shape, cylinder).
```

表示如果 X 在区域 0，那么它的形状必须是立方体或圆柱体。

3.2　环境表示

CLEVR-ASP 中的一个环境是由一组约束来定义的。每个约束决定了特定环境下物体的属性限制。数据集包含 30 个环境，每个环境中最多包含 15 个不同约束。部分约束的 ASP 编码表示以及对应表示含义见表3.1。

3.3 场景表示

CLEVR 数据集以场景图的形式表示场景，其节点表示使用其属性进行注释的对象，边表示对象之间的空间关系（前、后、左、右）。在 CLEVR-ASP 中，除了场景图表示之外，还在 ASP 中表示一个场景。下面以图 1 为例，展示部分场景的 ASP 表示：

```
%Objects in the scene
object(0). object(1). object(2). object(3).

%Attributes of objects
at(0, 2).
hasProperty(0, color, green).
hasProperty(0, size, large).
hasProperty(0, material, rubber).
hasProperty(0, shape, cylinder).
....

%Spatial relations between objects
front(1, 0). right(1, 0). ...
```

谓词 `object` 用于定义不同的物体（所有物体的名称用 0,1 等数字来表示），`hasProperty(Object, Attribute, Value)` 用于将对象的名为 Attribute 的属性的值设置为 Value。对象之间的空间关系用谓词 `left`、`right`、`front`、`behind` 来表示，例如 `left(A, B)` 表示 B 位于 A 的左侧。

3.4 图像生成

虽然 CLEVR 中的图像是通过随机采样的场景图生成的，但 CLEVR-ASP 生成的图像是基于已知符合环境约束的场景图。场景图的创建因此成为一个推理问题——在给定环境（基于答案集编程 ASP 的约束条件）和场景中预期物体数量 n 的前提下，需要完成以下任务：将每个物体分配到四个区域之一，并为颜色、尺寸、形状和材质属性赋值，确保这些属性符合环境中的约束条件。

解决这一问题，需要 ASP 求解器来完成。具体而言，ASP 求解器将返回一个解集，集合中的每个答案（即 n 个对象的一个一致属性值分配方案）代表一个场景图，或者说是场景中物体的一种可能配置。由于可能的配置方案众多，因此 CLEVR-ASP 随机采样了一百万个场景图用于后续的图像生成阶段。接下来，使用 Blender3 对场景图进行渲染，最终生成完整场景的图像。而部分场景的图像则是基于部分场景图生成的，具体做法是从实际场景图中随机移除一个物体，从而构造部分场景图。图3.1展示了场景图的构造过程。

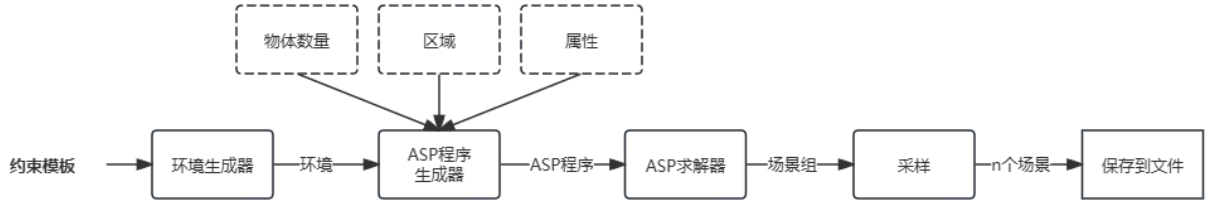


图 3.1: 生成环境以及该环境中的完整场景的流水线

3.5　问题表示

CLEVR-ASP 中的问题围绕部分场景中缺失的物体的颜色、大小、形状、材质这四个属性之一。具体的问题实例及 ASP 编码实例如下：

自然语言问题：与中等大小的红色物体的材质相同的，另一个圆柱体的颜色是什么？

```
query(Q) :- hasProperty(X, color, Q),
hasProperty(X, shape, cylinder),

hasProperty(Y, size, medium),
hasProperty(Y, color, red),
same_material(Y, X),
X != Y.
```

如果问题是关于属性 $A, A \in \{color, size, material, shape\}$，那么生成问题时，可能的解集 S 的基数为 $1 \leq |S| \leq |A|$，其中 $|A|$ 是属性 A 的所有可能取值集合的大小。例如，$|size| = 3 = \{large, medium, small\}$ 。如果生成的问题有 $|A|$ 个解，那么可判定该问题无效。例如，对于一个问物体尺寸的问题，答案是“尺寸可以为大或者中或者小”，显然这个答案是无效的，因为对于任意一个尺寸相关的问题，该答案均有效。各种类型的问题在个数上保持平衡。

3.6　问题生成

我们以 CLEVR 中的问题作为范本，来生成 CLEVR-ASP 中的问题。生成的问题均关于属性，避免生成是否类问题以及计数类问题。以下为问题模板样例：

```
What shape is the < Z2 > (size) < C2 > (color) < M2 > (material) [that is]
< R > (relation) the < Z > (size) < C > (color) < M > (material) < S > (
   ↪shape) ?
```

问题模板的实例化是基于相关图像的完整场景图完成的。感兴趣的对象始终是从完整场景中移除以生成部分场景图的对象。查询属性的选择需满足问题类型平衡的要求。查询对象的已知属性（填充上述模板中的 <Z2>、<C2> 或 <M2> 位置）是随机选取的。而 <R> 位置（即 left、right、front、behind 之一）的填充值也是随机选择的，但问题中的参考对象是根据完整场景中的空间关系选择的——具体而言，从与查询对象存在 <R> 关系的对象中选取一个。

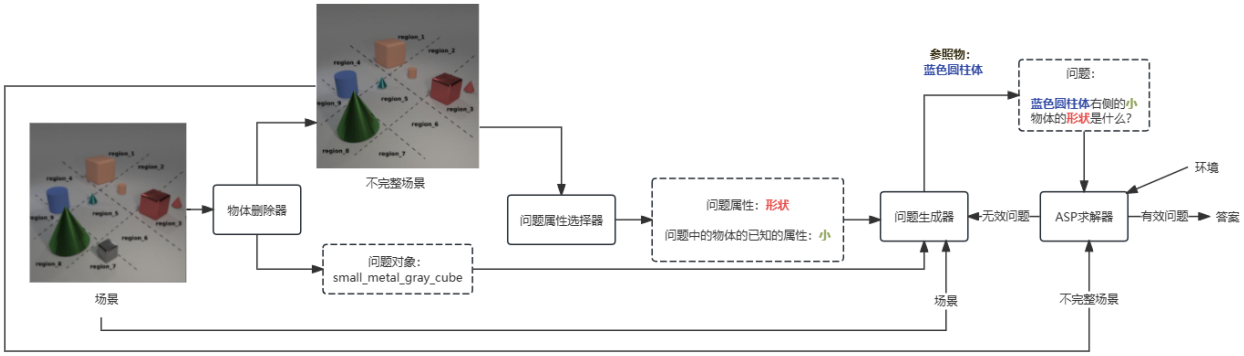


图 3.2: 生成部分场景和问题，并进行标记的流程

问题的 ASP 表示、部分场景以及环境中的约束都会被提供给 ASP 求解器，以确定问题的可能解。图3.2展示了问题生成的流水线过程。

3.7 本章小结

本章介绍了 CLEVR-ASP 数据集的构造过程，重点描述了如何基于完整场景图生成部分场景图，并通过 ASP（Answer Set Programming）实例化问题模板以确保问题的多样性和合理性。

首先，本章阐述了对象移除的原则，即如何选择查询对象以及如何保证其属性在问题类型上的均衡性。随后，详细说明了查询模板的填充策略，包括查询属性的选取、参考对象的确定以及空间关系的合理性，以确保生成的问题符合实际场景。最后，本章介绍了 ASP 求解器在问题生成过程中的作用，即利用 ASP 规则对部分场景进行推理，以确定查询属性的可能取值范围，从而生成符合逻辑约束的高质量视觉问答数据。

通过上述方法，CLEVR-ASP 数据集不仅保证了问题的可解释性，还增强了对复杂空间关系的推理能力，为视觉问答任务提供了更具挑战性的数据支持。

模板	描述
模板 1（取值约束）	`:- object(X), at(X, R), not hasProperty(X, P1, V1).` 解释: 对区域 R 中的所有物体，它们 P1 属性的取值均为 V1。 具体实现: : $-object(X), at(X, 0), nothasProperty(X, color, red).$
模板 2（否定约束）	`:- object(X), at(X, R), hasProperty(X, P1, V1).` 解释：对区域 R 中的所有物体，它们的 P1 属性的取值，均不能为 V1。 具体实现：:- object(X), at(X, 0), hasProperty(X, material, metal).
模板 3（恰有 N 个约束）	`:- #count{X: hasProperty(X, P1, V1), object(X), at(X, R)} != N.` **解释**：在区域 R 中，恰好有 N 个物体的 P1 属性的取值为 V1。 具体实现：:- #count{X: hasProperty(X, size, small), object(X), at(X, R')} != 2.
模板 4（至少有 N 个约束）	`:- #count{X1, X2: sameProperty(X1, X2, P1), object(X1), object(X2), at(X1, R1), at(X2, R2)} < N.` 解释：在区域 R1 和区域 R2 中，至少有 N 对物体，它们的 P1 属性的取值都是 V1。 具体实现：:- #count{X1, X2: sameProperty(X1, X2, shape), object(X1), object(X2), at(X1, 1), at(X2, 2)} < 1.
模板 5（或约束）	`:- object(X), at(X, R), not hasProperty(X, P1, V1), not hasProperty(X, P1, V2).` 解释：区域 R 中的所有对象都具有属性 P1 的 V1 值或属性 P2 的 V2 值。 具体实现：:- object(X), at(X, 1), not hasProperty(X, color, yellow), not hasProperty(X, color, blue).

表 3.1: 约束模板

第四章　实验设计

4.1　回答集编程

4.1.1　语法

项（terms）是 ASP 程序中最基本的元素，原子

4.1.2　语义

4.1.3　求解器

4.2　GLIP

4.3　大语言模型

第五章　问答系统的构建

5.1　回答集编程

5.1.1　语法

项（terms）是 ASP 程序中最基本的元素，原子

5.1.2　语义

5.1.3　求解器

5.2　GLIP

5.3　大语言模型

第六章　结论与展望

6.1　回答集编程

6.1.1　语法

项（terms）是 ASP 程序中最基本的元素，原子

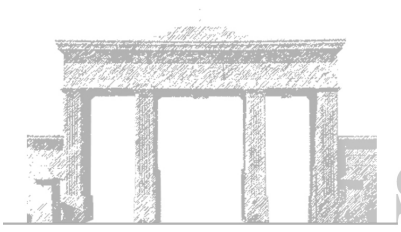
6.1.2　语义

6.1.3　求解器

6.2　GLIP

6.3　大语言模型

心求至善



SOUTHEAST UNIVERSITY