学校编码：10384　　分类号 T192 密级 公开
学号：23020081152XXX　　UDC 168

厦门大学

博 士 学 位 论 文

架构语言和逻辑的桥梁：面向超图的自然语言理解、生成、推理以及对话系统的关键技术研究

Between Language and Logic: Comprehension, Generation and Reasoning Using Systems of Hypergraph Transformations Embedded in an Integrated Cognitive Architecture

练 睿 婷

指导教师姓名：　周昌乐 教授
专 业 名 称：　人工智能基础
论文提交日期：　2015 年 4 月
论文答辩时间：　2015 年 6 月
学位授予日期：　　年　　月

答辩委员会主席: _____
评阅人: _____

2015 年 4 月

厦门大学学位论文原创性声明

本人呈交的学位论文是本人在导师指导下，独立完成的研究成果。本人在论文写作中参考其他个人或集体已经发表的研究成果，均在文中以适当方式明确标明，并符合法律规范和《厦门大学研究生学术活动规范（试行）》。

另外，该学位论文为（）课题（组）的研究成果，获得（）课题（组）经费或实验室的资助，在（　　）实验室完成。（请在以上括号内填写课题或课题组负责人或实验室名称，未有此项声明内容的，可以不作特别声明。）

声明人（签名）：

年　　月　　日

厦门大学学位论文著作权使用声明

本人同意厦门大学根据《中华人民共和国学位条例暂行实施办法》等规定保留和使用此学位论文，并向主管部门或其指定机构送交学位论文（包括纸质版和电子版），允许学位论文进入厦门大学图书馆及其数据库被查阅、借阅。本人同意厦门大学将学位论文加入全国博士、硕士学位论文共建单位数据库进行检索，将学位论文的标题和摘要汇编出版，采用影印、缩印或者其它方式合理复制学位论文。

本学位论文属于：

（　　）　1. 经厦门大学保密委员会审查核定的保密学位论文，于　　年　　月　　日解密，解密后适用上述授权。

（ ✓ ）　2. 不保密，适用上述授权。

（请在以上相应括号内打“✓或”填上相应内容。保密学位论文应是已经厦门大学保密委员会审定过的学位论文，未经厦门大学保密委员会审定的学位论文均为公开学位论文。此声明栏不填写的，默认为公开学位论文，均适用上述授权。）

声明人（签名）：

年　　月　　日

摘　要

一个能理解、生成自然语言并能用自然语言流利交流的系统将会有广泛的用途，但目前的计算语言学软件几乎在所有关键领域的表现，都与人类的需求相差甚远。本文在自然语言理解、生成和对话系统方面展开了深入研究，旨在搭建一个以能实现智能对话为长远目标的自然语言处理工具集。

本文的工作主要在认知体系结构 OpenCog 上进行的，OpenCog 是一个旨在实现通用人工智能（AGI）的集成软件平台，采用基于加权有向超图的知识表示体系 Atomspace，不仅为自然语言过度到逻辑语义形式提供了平台，而且方便了常识推理和自适应学习机制等研究工作的开展。

在理论层面上，本文的核心研究问题是设计一个面向英语的智能会话系统，其中包含自然语言理解（将英语句子转换成逻辑表达形式）、自然语言生成（将逻辑表达式转换成英语句子）、面向超图的推理以及目标驱动的对话控制。本文对该设计的实现和相关的实验工作主要集中在自然语言理解和生成方面，使其成功地从设计阶段走向能实现很多实用功能的阶段。在推理和对话控制方面，我们也进行了相关实验，并搭建了一个能集成自然语言理解、生成、推理以及对话控制等模块的原型系统。

在技术层面上，本文的研究工作在以下四个假设中展开：1）借助一个超图转换系统，使用依存关系语法、传统逻辑与谓词逻辑的合理结合，将各式各样的自然语言表达转换成满足下列要求的逻辑表达方式，是可行的。2）借助由超图转换表示的推理规则和基于超图表示的知识库，使用简单的逻辑推理，在上述自然语言理解框架输出的逻辑表达式上实现基本的常识推理，是可行的。3）借助一个超图转换系统和一个超图匹配系统，在由自然语言理解系统自动生成的二元组（语言表达式，逻辑表达式）组成的知识库中根据逻辑表达式找到匹配的语言表达式并生成自然语言，是可行的。4）利用上述的语言理解、生成和逻辑推理系统，构建一个有用且灵活的智能会话系统，是可行的。

在自然语言理解方面，本文论证了一个可行的自然语言理解流水线，它集成了改进后的卡耐基-梅隆大学的链语法解析器、改进后的 RelEx 关系抽取系统以及一个全新的 RelEx2Logic 系统，能将 RelEx 的输出转换成 OpenCog 所使用的概率逻辑网络形式的逻辑表达式。在自然语言生成方面，本文在 OpenCog 系统内实现了一个全新的自然语言生成系统，通过微观规划和表层生成，能将一个给定的逻辑表达式转换成相应的英文句子。在基于语言的推理方面，借助于 OpenCog 中的概念逻辑网络（PLN）框架，以基于上述自然语言理解框架输出的逻辑表达式作为推理前提，本文实现了不同形式的逻辑推理实例。最后，本文论证了上述的集成系统框架能被应用在自然语言问题系统和智能会话系统中。

本文阐述的研究工作目前仍在继续，距离达到人类水平的智能会话系统的长远目标，还有很长的路要走。但是，本文的理论研究工作和相关的实用工具的开发，已经向着我们的长远目标迈出了一大步，无论是被应用在现实世界上的相关自然语言软件，还是在 OpenCog 的推理系统上实现的基于自然语言的常识推理，都在自然语言处理领域有着深远的意义。

关键词： 自然语言理解；自然语言生成；知识表示；对话系统

Abstract

A software program that could fluently understand, generate, and converse in natural language would have a huge variety of uses. However, current computational linguistics software falls far short of human performance in almost every key area required for achieving this functionality, e.g. language comprehension, generation and dialogue. Our goal in undertaking the work reported in this thesis has been to build a conceptual and software toolset that will allow the construction of truly capable natural language processing software, with interactive dialogue as the main long-term application in mind.

We have carried out our work mainly within the OpenCog cognitive architecture, an integrated software platform designed with the goal of robust Artificial General Intelligence. OpenCog's key feature is the representation of knowledge in a weighted, labeled hypergraph store called the Atomspace, which allows representation of knowledge in a language that is both rigorous in the sense of probabilistic logic, and relatively simple and commonsensical and amenable to automated learning mechanisms.

On a theoretical level, the central problem we have confronted is the design of an English language dialogue system incorporating language comprehension (via mapping English sentences into logic expressions in the Atomspace format), language generation (via mapping Atomspace logic expressions into English sentences), logical reasoning on Atomspace logic expressions, and motivation-driven dialogue control. Our practical implementation and experimentation work has focused mainly on the comprehension and generation portions of the design, which we have successfully brought from the design phase to the stage of adequate initial practical functionality. We have also done some implementation and experimentation regarding the reasoning and dialogue control components, though our work there has been more at the research and early prototype level.

On a more technical level, regarding comprehension, generation and reasoning, we have

aimed to explore the following hypotheses: 1) That it is viable to transform a wide variety of natural language expressions into logic expressions via a system of hypergraph transformations, using dependency grammar and an appropriate combination of term logic and predicate logic; 2) That it is viable to carry out a variety of simple logical inferences, using inference rules represented as hypergraph transformations and representing aspects of human commonsense reasoning, based on combining the logic expressions output by the above-described natural language comprehension framework; 3) That it is viable to transform a wide variety of logic expressions into natural language expressions, via a process comprising a system of hypergraph transformations and centered on matching against a knowledge base composed of (linguistic expression, logic expression) pairs formed via automated language comprehension; 4) That a framework incorporating items 1-3 can be utilized for natural language dialogue.

In the comprehension domain, we have demonstrated a natural language comprehension pipeline that chains together an improved version of the Carnegie-Mellon Link Parser, an improved version of the RelEx relation extractor system, and a wholly new system called RelEx2Logic that maps the output of RelEx into logic expressions in the Probabilistic Logic Networks utilized by OpenCog. We have also demonstrated a novel natural language generation system, implemented inside the OpenCog system, which generates English sentences from a provided logic expression via a microplanning phase followed by a surface realization phase, where the surface realization process is based on matching fragment of the provided logic expression to fragments of logic expressions produced via comprehension using the above framework. We have demonstrated the use of OpenCog's Probabilistic Logic Networks (PLN) framework used to carry out a variety of logical inference examples based on premises that are logic expressions obtained by interpreting English sentences according to the above comprehension framework. Finally, we have shown that this integrated framework can be used for natural language question answering, a simple case of interactive dialogue.

The work reported in this thesis remains ongoing, and the long-term goal of a natural

language dialogue system with full human-level functionality remains a significant way off. However, the theoretical and practical tools created in the course of this thesis work constitute a significant step toward this long-term goal, and also possess significant value in themselves, both as tools for use in real-world software applications and as demonstrations of what kind of natural language processing it is possible to do using an OpenCog logic-based framework.

Keywords: Natural Language Comprehension; Knowledge Representation; Dialogue System; Natural Language Generation

目　录

Contents

第一章　绪论

目前，计算语言学软件几乎在每个关键领域都远远落后于人类的认知，如自然语言理解、生成和对话。计算语言学系统的功能在近几十年来得到稳步提高，在很大程度上是由于互联网的总体发展和相关的“大数据”现象所带来的语料库语言学的兴起。然而，在目前这么多研究方向中，哪一个最能带领该领域快速持续发展，还没有定论。

本文提出的结论，是遵循计算语言学功能发展的一个特定方向。本文工作是在一个旨在实现通用人工智能（AGI）的中集成认知体系中进行的，长远目标在于实现具有一定智能水平的自然语言理解、生成和对话系统。除此之外，本文也取得不依赖与该集成认知体系的具有价值的特定研究成果。

1.1　研究目的和意义

1.2　研究现状综述

近年来，自然语言处理(NLP)及其相关应用，尤其是方便人类和计算机交流的对话系统逐渐成为人们越来越关注的领域。它融合了计算机科学、语言学，以及其它学科。本文涉及了 NLP 领域中各个不同的研究方向，因此无法在短篇幅内给出每个方向的所有相关文献综述。Jurafsky’s 的书[?]已对这一领域进行了概述。在这一节中，我们针对本文研究密切相关的主要领域的研究现状进行大致的回顾，以及对具有可比性的研究方向和理论进行比较深入的探讨。

1.2.1 关于有效的可计算的语言处理的探索

XXX 这一小节留着备用。

从人工智能发展的早期开始，人们已认识到 NLP 在机器智能领域的关键性作用。在 1950 年，Alan Turing 发表了他的那篇经典文章“计算机与智能化（Computing Machinery and Intelligence）”。在文章中，他提出了一个理论，现在我们称其“作为智能标准的图灵测试”。他并没有称之为“自然语言处理”，但对于任何系统来说，要通过图灵测试，NLP 显然是至关重要的。

人工智能方面的先驱在早期过于乐观，在自然语言处理方面也是如此。1954 年，乔治城大学（Georgetown University）进行了一项试验，他们利用机器翻译将 60 多个俄语句子翻译成英文。这个试验的发起人声称：在 3 年或 5 年内，机器翻译将不成问题。然而，到 1966 年，一份报告显示：持续 10 年的研究都没有取得多少进展，而用于机器翻译的研究资金也大幅减少。事实上，在之后的 15-20 年里，机器翻译一直停滞不前，直到人们发现了完全不同的“统计语料库”分析法。通过基于规则的对话系统，人们取得了初步进展。例如聊天机器人心理医生 ELIZA[?]和用于受限“积木世界（blocks worlds）”的自然语言系统 SHRDLU[?]。图1.1中是 ELIZA 的对话例句。

尽管这些系统看起来很聪明，甚至有些像人类的思维，但它们显然缺乏对它们所使用的语言的深度理解，而且至今，类似的对话系统也没有比这些早期的原型系统懂的更多。

在 20 世纪 70 年代，为了克服早期这些基于规则的聊天机器人的缺陷，许多程序员开始编写“概念本体知识库”，这些知识库将真实世界的信息转换为计算机可以理解的数据，以便聊天机器人获得更多的知识[?]。相对来说，在 20 世纪 70 年代早期，基于知识本体的“问答”（QA）系统在限定的领域内是成功的，例如 LUNAR 系统。它能够回答那些对阿波罗登月任务[?]中带回的岩石进行地质分析的问题。(例如：高碱性岩石中铝元素的平均浓度是多少？)。1971 年的月球科学会议展示了 LUNAR 系统。当时，对于没有经过该领域训练的人提出的相关问题，LUNAR 能够答出 90%。在之后的几年间，更多基于限定领域的问答系统相继出现。

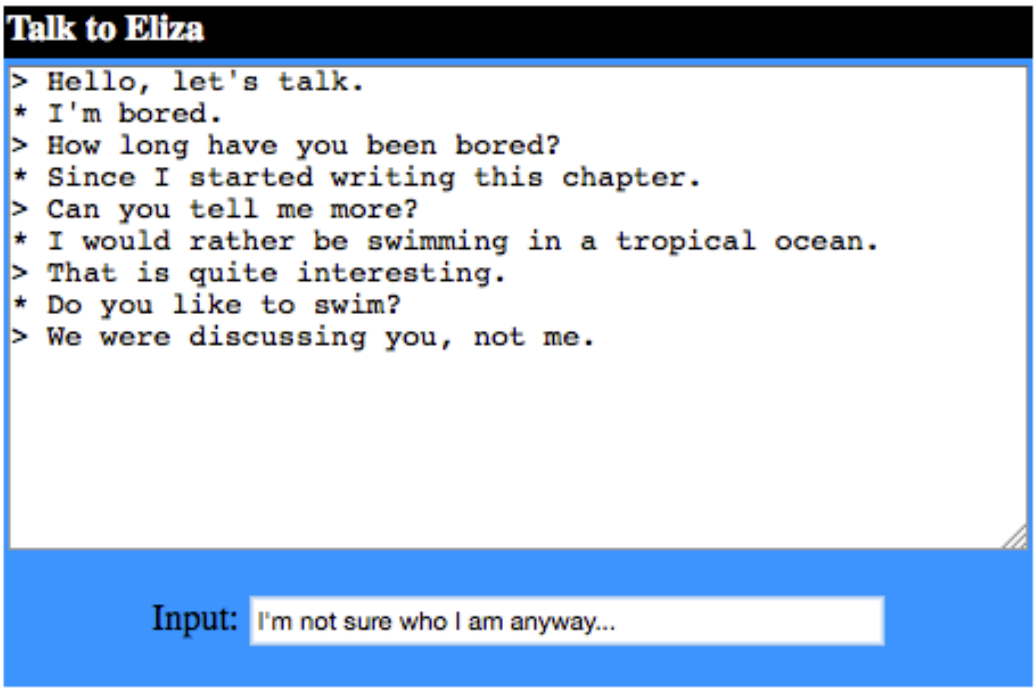


图 1.1　对话例句。这是与知名的聊天机器人心理医生 ELIZA 对话的现代版本。请注意：当“她”不知道该怎样回答的时候，“她”是如何转移话题的。

这些系统的共同特点是：都有核心数据库，或由相关领域的专家人工编写的知识系统。LUNAR 和当时其它一些系统的语言能力只能与 ELIZA 的语言能力相提并论。

经过 20 世纪 70 年代和 80 年代的发展，NLP 系统已能够融入了更先进的语言学理论，例如 20 世纪 80 年代末期，由加州伯克利大学 Robert Wilensky 教授开发的 Unix Consultant (UC)[?]。UC 以大量人工书写的知识库为基础，可以回答与 Unix 操作系统有关的问题。它会根据它对不同对话人所提出的问题的理解，组合出不同的答案。这种系统正是目前受限领域的自然语言问答系统（例如用于健康和生命科学的 EAGLi[?]）的前身。

在这一时期，理论语言学和实用计算 NLP 之间的关系是错综复杂的，甚至有些麻烦。主流的理论语言学主要是基于 Noam Chomsky 的理论，这个理论的中心是：假定句子的“表层结构”能微妙地转换成“深层句法结构”。尽管人们进行了大量尝试，但发现这些理论并不适合计算机应用[?]。

在 NLP 的发展历程中，最具革命性的事件是大型文本和语音语料库的出现，

以及统计分析方法的发展。基于统计的 NLP 的目标是通过识别和判定这些语料库中的各种模式，来实现语言处理功能。这些方法的流行主要是由于计算机和通讯技术的进步，以及人们越来越多地使用光盘，紧接着互联网，而并不是因为 NLP 本身的发展。由于人们对作为训练语料库的平行文本语料库的使用，并在训练语料库上使用相对简单的统计学习算法，例如基于隐马尔科夫模型(HMMs)[?]，机器翻译出现了重大进展。由 Chomsky 的早期导师 Zellig Harris[?]引领的标注语料库创建工作在众多实践中脱颖而出，同时一个有助于“有监督学习”任务的大规模分支领域出现了。例如：利用统计工具从标注语料库中学习语法，语料库中的每个句子都有人工标注的句法结构；或者使用人工标注的词性标注语料库，利用统计工具对新的文本进行自动词性标注。

“从大规模语料库中进行无监督学习”也是自然语言处理的研究重点(例如[?])，但事实证明这更加困难。而“半监督式学习”则是相对成功的。它融合了标注语料库和网络中大量非标记文本的使用[?][?]。

1.2.2 知识表示方法和逻辑推理研究综述

对于任何以实现复杂功能（如对话系统或复杂的问答系统）为目标的 NLP 系统来说，该系统如何表达内部知识是非常关键的。先进的 NLP 功能一般要求从多个句子中获得互相关的信息，这通常需要持续存储那些从多个句子中抽取的信息。这种存储机制必须能够支持相当灵活的语义知识操作。上节我们看到的例子是 FCG 使用的基于逻辑的语义表达形式。在本文的研究中，我们将使用一种不同的逻辑表达形式，即 OpenCog 中概率逻辑网 PLN（Probabilistic Logic Networks）的形式体系。

逻辑知识表示的优缺点

Strengths and Weaknesses of Logical Knowledge Representation

根据 Pei Wang[?]的理论，总体来说，逻辑推理系统通常包括以下组成部分：

- 一种形式化语言，能用于知识表示，以及系统与其环境之间的沟通。

- 一个语义系统，用于决定词的意义和句子真值。
- 一套推理规则，匹配问题和知识，能从前提推出结论等。
- 一个存储器，可以储存问题和知识，并提供推理的工作空间。
- 一个控制机制，负责选择前提和每一步推理中需要的推理法则。

前 3 个通常被认为是逻辑，或推理系统中的逻辑部分，最后的 2 个则被认为是推理系统中的控制部分。

使用逻辑来表达自然语言的语义，这涉及精确度和灵活度之间的平衡。逻辑是精确的，它的标志是“确定”。它带来一种用作定理证明的思考方式。它的优势在于稳定和系统的方式对表达式赋值并保持表达式的真值。在几乎没有歧义的技术领域，精确度是非常重要的，而逻辑显然是一个极好的架构。但是，当逻辑在意思和真值都比较模糊和模棱两可的领域中应用时，其适用性就不那么明显了，而且在人工智能和 NLP 领域中，逻辑的应用也常常引起争议。

进一步来说，逻辑领域中最长和最丰富的传统是以演绎推理为中心的。演绎推理是有限的推理形式，而且人们很少做关于自然语言的常识性推理，但仍然有大量的“归纳逻辑”[?][?]工作（包括最近人们重视的“概率归纳逻辑编程”[?]和溯因推理[?][?]）。这些推理方法也大量存在于我们所用的 PLN 系统中。

1.2.3　谓语逻辑 VS 传统逻辑

在数学和 NLP 环境下，最普遍的逻辑形式是“谓语逻辑”。它的独特之处是对变量的使用，这些变量可以被量化。两个常见的限量词是存在量词 ∃（“存在”）和普遍量词 ∀（“所有”）。在谓语逻辑公式中，变量可能是人们谈及的宇宙元素，也许是超越宇宙的关系或函数。举例来说：在标准的谓语逻辑中，一个句子，例如“Ravens are black”（乌鸦是黑的），它呈现的是个“一般命题”，如：

$$(\forall x)(Raven(x) \rightarrow Black(x)).$$

谓语逻辑通常与语义学的“理论模型”法一同出现，它视“逻辑公式”为特定领域的模型[?]。

另一个方法是“传统逻辑”推理法，这个概念实际上可以追溯到亚里士多德。在这里，“基本假设”是：命题由两个词语构成，推理过程反过来根据命题来构建。详解如下：

- 一个“词语”是言语表达的一部分，但就其本身而言，并没有对或错，例如“男人”或“凡人”。
- 一个“命题”包含两个词语，其中一个（“谓语”）是对其它词（“主语”）的“证实”或“否认”。它能够反应“真”或“假”。
- “三段论”是一种推理法，其中的一个命题（“结论”）必须遵循另外两个（“前提”）。

在“传统逻辑”推理法中，我们可以这样推理：

$$raven \rightarrow black$$

无需引入量词。

以下是一个标准的“三段论”逻辑推理示例：

$$A \rightarrow B$$

$$B \rightarrow C$$

$$\Rightarrow$$

$$A \rightarrow C$$

这也是演绎推理的简单形式。

在人工智能领域，Pei Wang 的 NARS 引擎运用了传统逻辑推理法[?]，并引入了独特的数学运算，用于管理与“传统逻辑关系”有关的不确定性。NARS 是建构在经验的基础上，而不是模型论语义学。

在许多方面，我们所使用的 PLN 逻辑形式化体系与 NARS 有类似的地方，但也存在巨大差异。PLN 在一个独特的数学框架下，同时利用传统逻辑和谓语逻辑。此外，PLN 还使用概率数学，以推导不确定真值的公式。反之，NARS 是基于原始的、非概率的、不确定的管理体系。PLN 也是以经验语义为基础，但形式与 NARS 不同。

图1.2展示了基本演绎推理、归纳推理和外展推理公式。这些公式是 PLN 和 NARS 共有的。在每个关系式右边的 $< s, c >$ 表示“每个关系的优势和信心”。PLN 和 NARS 使用不同的公式，从那些前提中推导（优势、信息）结论的真值。

deduction	induction	abduction
$M \rightarrow P <f_1, c_1>$ $S \rightarrow M <f_2, c_2>$	$M \rightarrow P <f_1, c_1>$ $M \rightarrow S <f_2, c_2>$	$P \rightarrow M <f_1, c_1>$ $S \rightarrow M <f_2, c_2>$
$S \rightarrow P <f, c>$	$S \rightarrow P <f, c>$	$S \rightarrow P <f, c>$

图 1.2　NARS/PLN term logic deduction, induction and abduction inference forms

1.2.4　Cyc 系统

在 NLP 系统中，也许开发最彻底的“谓语逻辑”是由 Cycorp 公司开发的商用系统 Cyc[?]。Cyc 系统拥有一个非常丰富的人工编码知识库，它的终极目标是：以谓语逻辑的形式，将所有人类常识进行编码。虽然它的知识库中已存有数百万的谓语逻辑公式，但到目前为止，似乎只对一小部分人类常识进行了编码。不过，作为智能应用系统，Cyc 已经相当成功了。

我们之所以在这里讨论 Cyc“知识表示”的基础，是为了与我们的系统进行比较。在 Cyc 系统中，概念名称被称作“常量”。在书写时，它们以”#$”开始。以下是几种常量：

- 单个词汇被称为“个体词”，例如：#$BillClinton (比尔 · 克林顿或#$France（法国）。

- 集合词，例如 #$Tree（树）-ThePlant（植物）(包括所有的树) 或#$EquivalenceRelation（等价关系）(包括所有的等价关系)。

- 真值函数，可以运用到一个或多个概念中，并反馈“真”或“假”。例如：#$siblings is the sibling relationship，如果两个参数是 siblings，那就是真的。按照惯例，真值函数以小写字母开头。真值函数可能会被分拆为逻辑连接词（如：#$and, #$or, #$not, #$implies）和量词（如：#$forAll, #$thereExists 等）。

- “函数”能够从给定的词中生成新词汇，例如：#$FruitFn，当提供了一个描述某种植物类型（或集合）的参数，它会给出这类植物的一组水果。按照惯例，“函数常量”以“大写字母”开始，以字符“Fn”结束。

“常量”之间由谓语联系在一起。最重要的谓语是：#$isa 和#$genls。第一个(#$isa)描述的是：某个个体是某个集合中的一个例子（如：specialization）；第二个（#$genls）描述的是：一个集合是另一个集合的子集合（例如：generalization）。利用特定的 CycL 句子，我们可以得出概念事实。谓语是写在它的参数之前的，在圆括号内。例如：

```
(\#\$isa \#\$BillClinton \#\$UnitedStatesPresident) \;
```

意思是”Bill Clinton belongs to the collection of U.S. presidents” ;

```
(\#\$genls \#\$Tree-ThePlant \#\$Plant) \;
```

意思是”All trees are plants”; and

```
(\#\$capitalCity \#\$France \#\$Paris) \;
```

意思是”Paris is the capital of France.”

下面是比较封复杂的例子，它表示了一组或一类词的规则，而不是任意特定的个别词：

```
(\#\$relationAllExists \#\$biologicalMother
\#\$ChordataPhylum \#\$FemaleAnimal)
```

Cyc 知识库被分为多个“微理论”库，它们是概念和事实的集合，每个“微理论”库都与一个特定领域相关联。每个“微理论”库都不能有相互矛盾的信息，而且可以通过 Bayesian 网络提供概率真值。

Cyc 配备了一个复杂的、基于短语结构语法的 NLP 系统，这个系统将自然语言句子映射为 Cyc 逻辑形式。由于这个系统本身的特性，我们尚不清楚它到底具有什么样的优点和缺点。

1.2.5　ConceptNet

知识表示的另一个模式是由 ConceptNet 提供的[?]。它从 Cyc 系统和形式逻辑中借鉴了一些理念，但没有考虑复杂的位元，如量词,并且比较接近自然语言层面。ConceptNet 是个大规模的语义网络，它表达概念（由单词或短语表述）之间的关系。图1.3是关于子网络的例子。

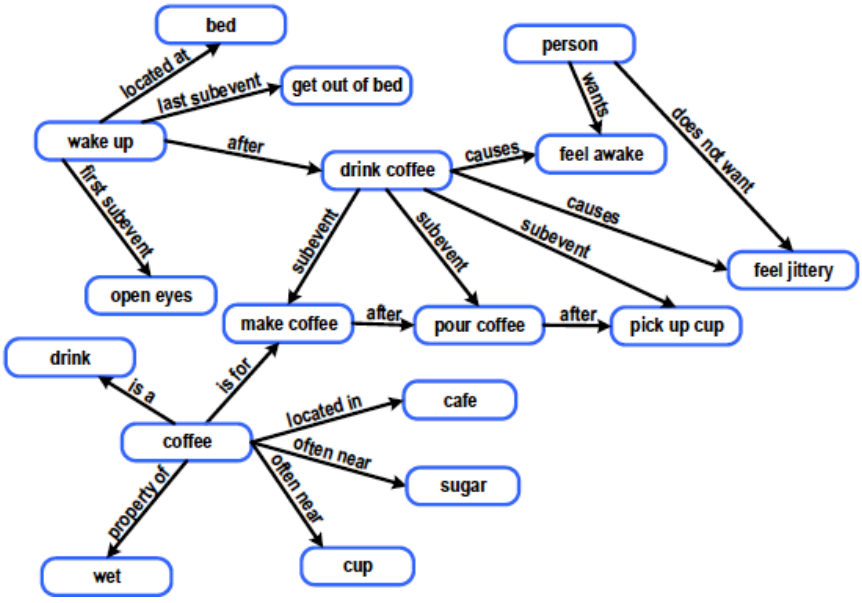


图 1.3　An illustrative fragment of ConceptNet

ConceptNet 系统的节点是自然语言碎片，这些碎片按照某个句法模式的本体进行了半结构化。它们适合一阶概念（作为名词短语，如 potato chips）和二阶概念（作为动词短语，如：buy potato chips）。

节点可以由 19 个语义关系相连接:

- 事物
  - IsA ?(corresponds loosely to hypernym in WordNet)
  - PropertyOf ?(e.g. (PropertyOf “apple” “healthy”))
  - PartOf ?(corresponds loosely to holonym in WordNet)
  - MadeOf ?(e.g. (MadeOf “bottle” “plastic”))
- 事件
  - FirstSubeventOf, LastSubeventOf ?(e.g. (FirstSubeventOf “act in play” “learn script”))
  - EventForGoalEvent ?(e.g. (EventForGoalEvent “drive to grocery store” “buy food”))
  - EventForGoalState ?(e.g. (EventForGoalState “meditate” “enlightenment”))
  - EventRequiresObject ?(e.g. (EventRequiresObject “apply for job” “resume”))
- 动作
  - EffectOf ?(e.g. (EffectOf “commit perjury” “go to jail”))
  - EffectOfIsState ?(e.g. (EffectOfIsState “commit perjury” “criminal prosecution”))
  - CapableOf ?(e.g. (CapableOf “police officer” “make arrest”))
- 空间
  - OftenNear ?(e.g. (OftenNear “sailboat” “marina”))
  - LocationOf ?(e.g. (LocationOf “money” “in bank account”))
- 目标
  - DesiresEvent, DesiresNotEvent ?(e.g. (DesiresEvent “child” “be loved”))

- 功能
  - UsedFor ?(e.g. (UsedFor “whistle” “attract attention”))
- 通用
  - CanDo ?(e.g. (CanDo “ball” “bounce”))
  - ConceptuallyRelatedTo ? (e.g. (ConceptuallyRelatedTo “wedding” “bride” “groom”)

OpenCog 的 Atomspace 是本文的研究重点，它与 ConceptNet 有些共同之处。我们视 Atomspace 为“加权标记超级图”，但 ConceptNet 只是一个“加权标记图”。此外，在可以直接表示的关系复杂性方面，Atomspace 与 ConceptNet 有所不同。Atomspace 包含与 ConceptNet 相似的简单节点和链接，但还有更加复杂的，且能够表示量词关系，还含有可执行程序等。它的设计原则是：先利用简单的、ConceptNet 类的方法，尽可能地表示，但之后在必要的情况下使用更复杂的表示工具。大多数“表示工具”都来自传统逻辑，而不是谓语逻辑，但必要时也使用与“明确量词关系”相对应的节点和链接。

1.2.6　言语行为理论及相关研究综述

言语行为理论

Speech Act Theory 分析“对话系统”需要生成的各类话语的一个方法是：使用由 Austin[?]和 Searle[?] 开创，又经他人进一步完善的“言语行为论”。本文所阐述的对话研究中使用的正是这种方法。我们在这里先回顾非常基本的言语行为论，然后介绍言语行为论的具体变化。在研究过程中，我们发现这个理论是极有用的。

言语行为论的核心概念是：分析不连续言语行为的语言学行为，以实现特定的目标。在 OpenCog 环境下，这是一种使用方便的理论方法，因为它促使我们像

对待 OPenCog 系统实施的其它行为一样对待言语行为，并推动我们通过标准的 OpenCog 行为选择机制来处理言语行为。

- **言语行为论**：探讨那些可以通过言语来表现的行为类型。
- **言表意的行为**：某句话的外在表现，例如：真实言语和它的表面含义。
- **言外行为**：言语的“言外之力”，例如它的目标含义（作为社交层面的有效言语行为）。（按照 Searle 的使用方法，“言语行为”有时仅仅指的是言外行为。我们不用这个方法。）
- **言后行为**：它的实际效果，例如：说服、劝说、吓唬、启发、激励，或者让某人做或实现某个事项，不论是有意或无意的。

在 Searle 看来，说话人通过他们的言语，只能获得 5 类言外之力，分别是：

- 断言类：说话人自己承诺事情是真的。The sky is blue（天是蓝色的）
- 指令类：说话人试图让受话人做某事。Clean your room!（打扫你的房间！）
- 承诺类：说话人对未来的行为做出承诺。I will do it (我将会做这件事)
- 表达类：说话者表达了某些心理状态。I’m Sorry (对不起)。
- 声明类：说话人带来了不同的状况。The meeting is adjourned.（这个会议休会了）。

受这个本体论的启发，Twitchell 和 Nunamake 在他们 2004 年的论文（标题为：言语行为归档：分析持续谈话和其参与者的概率统计法。[?]）中提出了更加精细的“42 种言语行为”本体论，叫做 SWBD-DAMSL (DAMSL = Dialogue Act Markup in Several Layers）。尽管有少量不符合 Searle 的观点，并被分类为“其它”，但几乎他们所有的 42 种言语行为类型都能够被映射到 Searle 的 5 个高级类别之一。图1.4和1.5描述了这 42 种行为和它们与 Searle 分类之间的关系。我们已经利用了这个 Searle 解析法，它给我们正在进行的自然语言对话研究带来了灵感。

Tag Name	Tag	Example
STATEMENT-NON-OPINION	sd	Me, I'm in the legal department.
ACKNOWLEDGE (BACKCHANNEL)	b	Uh-huh.
STATEMENT-OPINION	sv	I think it's great.
AGREE/ACCEPT	aa	That's exactly it.
ABANDONED, TURN-EXIT OR UNINTERPRETABLE	%	So,-
APPRECIATION	ba	I can imagine.
YES-NO-QUESTION	qy	Do you have to have any special training?
NON-VERBAL	x	[Laughter], [Throat-clearing]
YES ANSWERS	ny	Yes.
CONVENTIONAL-CLOSING	fc	Well, it's been nice talking to you.
WH-QUESTION	qw	Well, how old are you?
NO ANSWERS	nn	No.
RESPONSE ACKNOWLEDGEMENT	bk	Oh, okay.
HEDGE	h	I don't know if I'm making any sense or not.
DECLARATIVE YES-NO-QUESTION	qyCd	So you can afford to get a house?
OTHER	other	Well give me a break, you know.
BACKCHANNEL IN QUESTION FORM	bh	Is that right?
QUOTATION	Cq	You can't be pregnant and have cats.
SUMMARIZE/REFORMULATE	bf	Oh, you mean you switched schools for the kids.
AFFIRMATIVE NON-YES ANSWERS	na	It is.
ACTION-DIRECTIVE	ad	Why don't you go first
COLLABORATIVE COMPLETION	C2	Who aren't contributing.
REPEAT-PHRASE	bCm	Oh, fajitas
OPEN-QUESTION	qo	How about you?
RHETORICAL-QUESTIONS	qh	Who would steal a newspaper?
HOLD BEFORE ANSWER/AGREEMENT	Ch	I'm drawing a blank.
REJECT	ar	Well, no
NEGATIVE NON-NO ANSWERS	ng	Uh, not a whole lot.
SIGNAL-NON-UNDERSTANDING	br	Excuse me?
OTHER ANSWERS	no	I don't know
CONVENTIONAL-OPENING	fp	How are you?
OR-CLAUSE	qrr	or is it more of a company?
DISPREFERRED ANSWERS	arp	Well, not so much that.
3RD-PARTY-TALK	t3	My goodness, Diane, get down from there.
OFFERS, OPTIONS COMMITS	commits	I'll have to check that out
SELF-TALK	t1	What's the word I'm looking for
DOWNPLAYER	bd	That's all right.
MAYBE/ACCEPT-PART	aap	Something like that
TAG-QUESTION	Cg	Right?
DECLARATIVE WH-QUESTION	qwCd	You are what kind of buff?
APOLOGY	fa	I'm sorry.
THANKING	ft	Hey thanks a lot

图 1.4　The 42 speech acts presented in the SWBD-DAMSL study

言语行为分类

Speech Act Classification XXX 包括由经验数据衍生的言语行为分类

1.2.7　对话系统和问答系统研究综述

对话系统

Dialogue Systems

Turing（图灵）曾在其 1950 年的论文中阐述了他对终极 NLP 应用的预期：对话系统能够以自然的方式与人类使用者进行谈话。粗糙的对话系统（如：Apple 的 SIRI），在今天已成为我们日常生活的一部分，但真正具有人类思维能力的对话系

Assertives	Expressives	Directives
STATEMENT	OPINION	YES-NO-QUESTION
YES ANSWERS	ABANDONED/UNINTERPRETABLE	WH-QUESTION
NO ANSWERS	BACKCHANNEL/ACKNOWLEDGE	DECLARATIVE YES-NO-QUESTION
QUOTATION	RESPONSE ACKNOWLEDGEMENT	BACKCHANNEL-QUESTION
AFFIRMATIVE NON-YES ANSWERS	SIGNAL-NON-UNDERSTANDING	SUMMARIZE/REFORMULATE
COLLABORATIVE COMPLETION	AGREEMENT/ACCEPT	ACTION-DIRECTIVE
RHETORICAL-QUESTIONS	APPRECIATION	OPEN-QUESTION
NEGATIVE NON-NO ANSWERS	CONVENTIONAL-CLOSING	TAG-QUESTION
OTHER ANSWERS	HEDGE	DECLARATIVE WH-QUESTION
OR-CLAUSE	HOLD BEFORE ANSWER/AGREEMENT	
DISPREFERRED ANSWERS	REJECT	Other
	CONVENTIONAL-OPENING	OTHER
Commissives	DOWNPLAYER	THIRD-PARTY TALK
OFFERS, OPTIONS, & COMMITS	MAYBE/ACCEPT-PART	NONVERBAL
	APOLOGY	
Declarations[1]	THANKING	
	REPEAT-PHRASE	

[1]None of the 42 dialogue acts could be described as a declaration

图 1.5 Correlation of the 42 SWBD-DAMSL speech acts with Searle's high-level speech act categories

统仍然没有实现其研究目标。

图1.7描述了对话系统的通用结构[?]。除了 NL 生成和理解，以及从各种资源中获取相关知识，其主要模块是对话管理器。这个模块决定在谈话中的每个节点说什么。

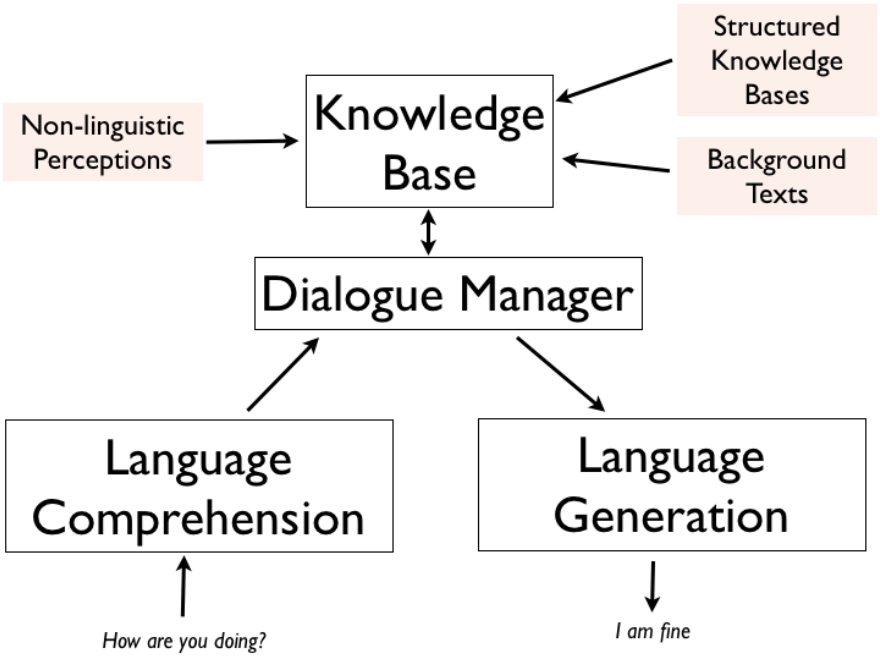


图 1.6 A generic dialogue-system architecture

总之，对话管理器全方位管理对话。它采用用户文本的语义表示、确定文本是否契合上下文，并构建系统响应的语义表示。一般来说，在它的职责中，以下几个

是必要的:

- 保存对话历史
- 采用一定的对话策略
- 处理格式错误和无法识别的文本
- 检索文件或数据库中的内容
- 确保为使用者提供最好的响应
- 管理启动和系统响应
- 处理语言学问题
- 言语分析
- 以任何可用的、相关的非语言信息来构建语言结构。

目前大多数对话系统都利用一小部分固定规则来处理对话管理（这些规则代表人工制定的言语策略）。近年的一些系统还提供另一种方法：利用概率模型（如：POMDP）来控制对话，但这些系统还不具备有价值的实用性[?][?]。

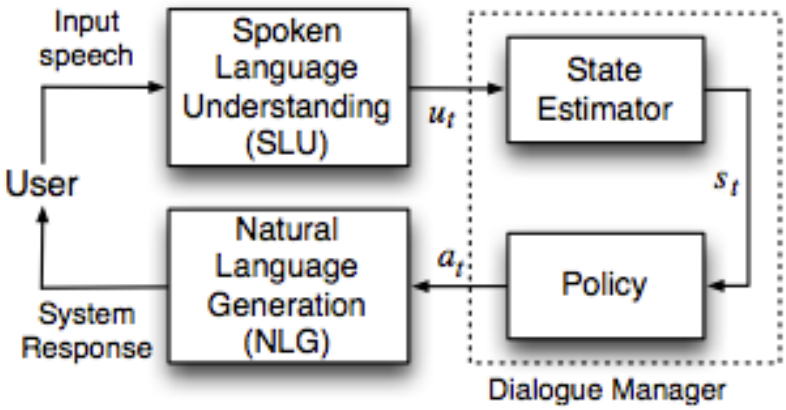


图 1.7　A POMDP-based reinforcement-learning-driven dialogue-system architecture

问答系统

Question Answering

就连接自然语言理解、生成和知识表示而言，也许最简单、最有意义的模式就是“问答系统”。问答(QA)系统是一种特殊形式的对话系统，它让使用者以自然语言提问，然后以自然语言响应。

许多现代 QA 系统基本上都是以文档为驱动的信息检索系统，请见图1.8中的示例[?]。这种系统（其中的问题被直接提交至一个大规模文本语料库）的表现通常优于 20 世纪 70 年代的早期 QA 系统（依赖人工编码的知识库）。

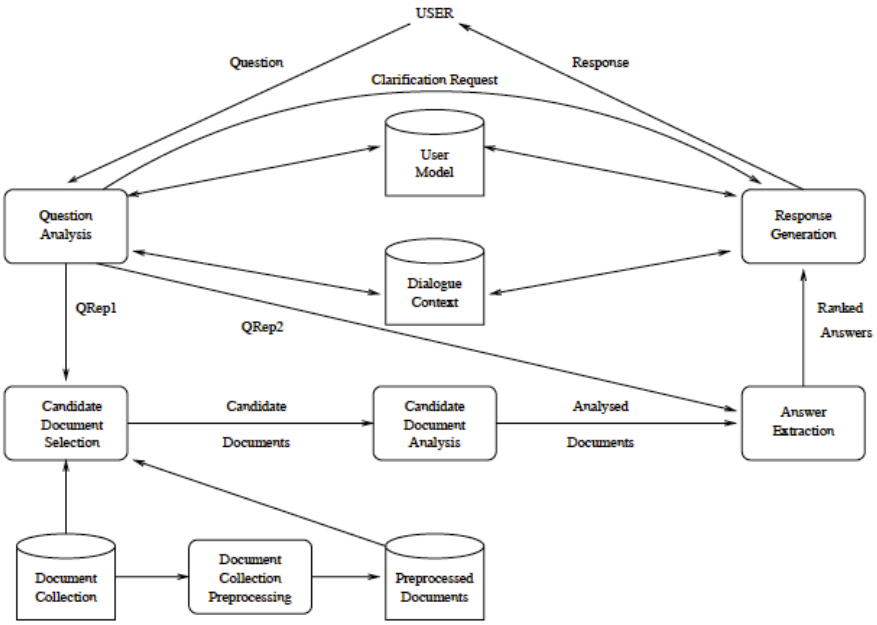


图 1.8 典型的文件驱动问答系统的架构

这种以文件为驱动的 QA 系统通常包括一个问题分类模块。这个模块确定问题和回答的类型。分析问题之后，系统一般会运用几个模块。文本的数量逐步减少，这些模块则越来越多地应用于复杂的 NLP 技术。因此，文件检索模块可以使用搜索引擎来识别那些可能含有“答案”的文件，或文档中的段落。随后，一个过滤器预先挑选出包含同类字符的小文本碎片，用作预期回答。例如：如果问题是“谁发明了青霉素？”，过滤器会反馈包含人名的文本。最后，一个“回答抽取模块”会在文本中进一步寻找线索，以确定候选的答案是否能正确地回答这个问题。

另一方面，如果问答系统会根据对文本知识的深层理解来做出响应，那么“快速处理文本来响应问题”就不是一个非常可行的策略。更准确地说，用于为 QA 系统提供知识的文件必须预先由一个 NL 理解系统彻底地“读”和理解。后面的那个方法是我们要在本文的第**??**章中探讨的。我们开发了一个简单的 QA 系统，它是一个整体互动对话体系的一部分。

在 2010 年，IBM 的问答系统 Watson 以极大的优势击败了另两个 Jeopardy 冠军奖的获得者：Brad Rutter 和 Ken Jennings。Watson 综合利用了信息检索法和基于推理的先进方法[?]，可以说是迄今为止最先进的问答系统。

问答系统的关键问题

Key Issues Regarding Question Answering

问答系统是一个复杂的探索课题，它涉及众多问题。举例来说，有证据表明：某种问题要难于其它问题。询问“为什么”和“怎样”的问题往往比询问“是什么”和“在哪里”的问题更难回答，这是因为它们要求对因果关系，或 instrumental 关系的理解。这些关系通常由分句或独立的句子来表达[?]。

在 2002 年，一组研究人员绘制了一幅关于问答系统的研究路线图[?]。他们当时提出的问题在今天仍然与我们息息相关。以下是他们发现的问题：

- 问题类别：不同类型的问题（例如“列支敦士登的首都是哪里？”VS“为什么会形成彩虹？”VS“玛丽莲 · 梦露和加里 · 格兰特出演过同一部电影吗？”）要求使用不同的策略来发现答案。
- 问题处理：相同的信息可能是用不同的方式来表达的，有些是疑问句（“莱索托国的国王是谁？”），有些是祈使句（“告诉我莱索托国王的名字。”）。因此，梳理出有效信息需要花些功夫。
- 上下文和问答：上下文可能用于厘清某个问题、解决歧义，或者追踪一系列问题的调查。(例如“为什么 Joe Biden2010 年访问了伊拉克？”。这个问题也

可能这样问：为什么副总统 Biden 去访问，而不是 Obama 总统；为什么他去的是伊拉克，而不是阿富汗或其他国家；为什么他是 2010 年去的，而不是在那之前或之后；或者 Biden 希望在那次访问中取得什么成果等。)

- 回答公式：QA 系统生成的结果以尽可能自然的方式呈现。
- 实时问答：不论问题有多复杂，“快速回答”在实际应用中都非常重要。
- 互动问答：经常出现的一种情况是：QA 系统没有很好地获取信息需求，因为问题处理部分可能没有成功地对需要抽取的问题或信息进行正确分类，而且生成答案也不容易检索。在这种情况下，提问的人可能不仅想重新表达问题，还想与系统进行对话。此外，系统也可以利用之前回答过的问题。
- 先进的 QA 推理：更多复杂的问题等待着书面文本或结构数据库以外的回答。为了完善 QA 系统，在其中添加这些功能，以下几项工作是必要的：整合推理模块、建立多种知识库、对通用知识，常识推理机制，以及不同领域的特定知识进行编码。我们的研究正是要推动这方面的发展。
- QA 用户归档：用户归纳是用于获取提问者的数据，包括上下文数据、兴趣、提问者常用的推理方案、系统和使用者之间不同对话的共同点等。这种归档可以为 QA 系统的表现提供有价值的指引。

另一个与 QA 有关的问题是：如何判断某个“回答”的质量。以下是几个常用标准：

- 相关性：“回答”应该是对问题的响应。
- 正确性：“回答”应该符合事实。
- 简洁性：“回答”不应该包括无关信息。
- 完整性：“回答”应该完整，例如：不完整的回答不应该得满分。
- 连贯性：“回答”应该是连贯的，这样才能方便提问者阅读。

- 正当理由:“回答”中应该带有足够的上下文，以便提问者了解为什么选择了这个答案。

到目前为止，大多数研究都关注了“相关性”。

1.2.8　自然语言理解、生成研究综述

自然语言理解是 NLP 的分支领域，它是通过使用软件来理解自然语言的意义（语音，或者本文更关注的“文本”）。自然语言理解主要是将自然语言转换成抽象的语义形式，以获取语言的含义；或者是转换为某种响应（例如对某个问题的回答），以说明其理解了语言的含义。

自然语言理解

Natural Language Understanding XXX 语言需要调整改进 XXX 目前主流的自然语言理解主要是通过以下流程逐步完成：词法分析、句法分析、语义关系抽取、语义理解。词法分析主要是 XXXX，并不是本文的研究重点，因此这里从其他几方面来介绍各流程的研究现状。

1. 句法分析Syntactic Parsing

   通常自然语言理解是通过对句子的某些，或全部句法法进行分析来完成的。大量的形式化体系和算法通过它们自己的独特方式对句子的句法结构进行解析。大体来说，这些可以归类为“依存语法（DG）”对“短语结构语法（PSG）”。PSG 首先对句子进行短语分析，然后指出单词和短语之间的关系，以及短语之间的关系；另一方面，DG 仅仅在句子中的单词之间标注（有标记的）连接关系。后者（DG）是我们要在本文中探讨的语法类型。

   这里的“依存”是指语言单位（如单词）由有向链接相互联系在一起。一般来说，在 DG 中，（限定）动词被视为句子或子句结构的中心，其它所有句法单位

（单词）是直接或间接地与动词通过有向链接相连。这种有向链接被称为“依存”。句子结构是由一个词（中心词）与它的依存词之间的关系而决定的。

“依存”关系是一对一的对应关系：对于句子中的每个元素（例如：单词或语素）来说，实际上句子中都有一个与其相对应的节点。这种一对一的对应关系决定了“依存语法”就是单词（或语素）的语法。所有的句子都有元素和将元素组成结构的依存关系，这种情况应该与短语结构语法的“成分关系”进行比较，“成分关系”是一对一，或一对多的对应关系，也就是说，对于句子中的每个元素来说，有一个或多个与其对应的节点。这种不同带来的结果是：相比短语结构，依存结构非常紧凑，因为它往往包含很多小的节点。从计算机处理的角度来说，这种简洁的结构是有益的。

2. 关系抽取Relationship Extraction 自然语言理解中的一个重要方向是**关系抽取**。在 NLP 领域中，关系抽取已成为一个重要的研究方向，一方面是因为它的实际应用价值，另一方面因为它被看做是语义分析的一部分，而且是目前的技术相对容易实现的那部分。到目前为止，吸引最多注意力的关系抽取是命名实体之间的关系识别，例如：“个人从属关系”和“组织地址关系”。

   一般来说，关系可以由一个元组的形式来定义的，$t = (e_1, e_2, ..., e_n)$。在这里，e_i 是文本中预定义关系 r 的实体。大多数关系抽取系统主要关注二元关系的抽取。例如：

   ```
   位于（厦门，中国）
   father-of(Richard Li, Li Ka-Shing)
   ```

   抽取“高阶关系”也非常有意义。例如这个句子：“At codons 12, the occurence of point mutations from G to T were observed”（“在密码子 12，观察到从 G 到 T 的点突变”）。句子中出现了一个 4 进制生物医学关系，可以描述为：

   ```
   point mutation(codon, 12, G, T)
   ```

   目前人们普遍视“关系抽取”为一个有监督分类问题，它从一个语料库开始（语料库中包含由人类标记的相关语义关系），然后利用统计方法来对标记的关系建模，并学习出适合应用于其它文本的统计模型。

目前，关系抽取系统的主要限制在句子层面上运用。事实上，关系可能跨越句子，甚至贯穿不同的文件。然而，解决这一问题需要一定的常识和非常灵活的知识表示。本文的研究也无法直接解决这一问题，但我们相信：通过将句子意义映射为通用的知识表示，我们已经奠定了解决问题的基础。在这种知识表示中，跨句或跨文本的通用联系和推理是可以被实现的。

3. 流结构语法

将流结构语法（FCG）与本文使用的语言形式化体系进行比较分析是很有意义的。在 FCG 中，一句话语的信息是以语义和句法结构组织在一起的。语义结构是对“话语意思”的分解，它含有特定语言的语义分类，例如：“put”事项被归类为“起因-移动-位置”类事项，包括一个施事者（agent）、一个受事者（patient）和一个位置（location）。句法结构是将话语形式分解为成分和词素，还包含附加的句法分类，比如：句法特征（例如：数量和性别）、词序限制等。

从理论上来说，FCG 是基于一种程序性语义的方法，在这个意义上，话语的意思是听话人假定要执行的程序，因此，概念化便是一个规划过程（规划该程序），而解析便是对该程序的执行过程。FCG 中,有关配对句法和语义结构的例子（短语“the ball”）,请见图1.9 and 1.10。

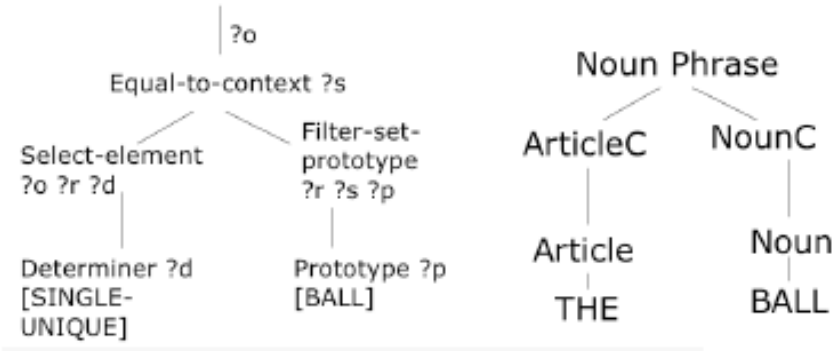


图 1.9　FCG syntactic and semantic structure for ”the ball”

所有的 FCG 规则都是双向的。通常在产生过程中，所要表达的语义内容是与语义结构相统一的，有可能产生一组绑定。如果成功了，绑定会与语义结构

```
((unit-1
    (SEM-SUBUNITS (unit-3)))
 (unit-3
    (CONTEXT ((LINK ?s)))
    (MEANING ((EXTERNAL-CONTEXT ?s)))
    (SEM-SUBUNITS (unit-4 unit-5)))
 (unit-4
    (CONTEXT ((LINK ?o ?r)))
    (MEANING ((SELECT-ELEMENT ?o ?r ?d)))
    (SEM-SUBUNITS (unit-6)))
 (unit-5
    (CONTEXT ((LINK ?r ?s)))
    (MEANING ((FILTER-SET-PROTOTYPE ?r ?s ?p)))
    (SEM-SUBUNITS (unit-7)))
 (unit-6
    (CONTEXT ((LINK ?d)))
    (MEANING ((DETERMINER ?d [SINGLE-UNIQUE]))))
 (unit-7
    (CONTEXT ((LINK ?p)))
    (MEANING ((PROTOTYPE ?p [BALL])))))
```

图 1.10 FCG semantic structure for ”the ball” in list format

相融合。这种融合可以理解为“部分统一”，但它利用结构中那些遗漏部分扩展了结构。在句法分析过程中，被分析的句子与句法结构是统一的，同时，结果中的某些部分被添加到语义结构中。

这篇论文所提出的形式化体系在概念上有些类似 FCG。此外，我们有配对句法和语义结构，而且还进行双向处理。在第5章中，我们将论述链语法，它将句子转换成句法结构，以及 RelEx 和 RelEx2Logic 模块，它们将句法结构转换成语义结构。在第6章中，我们将论述 Microplanner 和 SuReal 模块，从另一个方向，将语义结构转换成句法结构，再生成句子。OpenCog 中的模式匹配器（Pattern Matcher）使用我们的句法和语义结构时，也将这些结构视为有效的程序，同样实现了“程序性语义”。

我们的形式化体系与 FCG 的着重点不同。FCG 主要是用作探索问题的理论工具，而我们所做的 OpenCog 系统则是用于真实世界的实际应用。

自然语言生成

Natural Language Generation

自然语言生成（NLG）是 NL 的反向过程。NLG 是从信息的计算表示中自动生成人类（自然的）语言。大体上，NLG 系统与以下问题有关：

- What should I say?（我应该说什么？）
- How should I say it?（我应该怎么说？）

这涉及许多相互关联的计划过程，如：

- 决定要说的信息
- 构建语篇计划
- 将信息块转换为语篇单位
- 选择适当的短语和单词
- 输出正确的语法

将这个流程分解为阶段的方法如下：

(a) 宏观计划

(b) 微观计划

(c) 表层实现

宏观计划涉及选择和组织内容。它输入的是一个或多个沟通目标：解释、描述，或提问；引起听众的某种行动或思考等。宏观计划输出的是一种知识架构，这个架构不一定是语言的属性，而是体现智能体所要沟通的信息。除了一般性内容，这种知识架构可能包含一些与沟通过程有关的信息，例如：不同的知识块应该以什么样的顺序来传达。

有些宏观计划法涉及语篇结构的特定理论，例如：修辞结构理论[?]。在本文所介绍的研究中，我们采用了宏观计划，利用名为“OpenPsi”的“动机认知模型”(这个模型的构建密切遵循人类心理学)。

微观计划则运用知识架构，并将它们分为句块。微观计划必须处理多种语言问题，例如：

- 句子范围：是否可以将两片叶子接在一起，怎样接在一起。(“我今天饿了。我去了 Burger King。”VS“我今天饿了,所以我去了 Burger King。”)
- 代词化
- 聚合：删除重复内容。(“抽烟对你不好。抽烟会缩短你的寿命。抽烟让你口气不佳”VS“抽烟对你不好、缩短你的寿命，而且让你口气不佳”)
- 主题化
- 主题排序

到目前为止，我们的大多数微观计划都是为特定的应用而专门制定的。有一个名为 SPUD[?]的多用途微观计划框架，它利用的是一个基于逻辑的计划流程形式。尽管在细节上有所不同（原来的 SPUD 是基于树-邻近语法），但我们所描述的微观计划在概念上受到了 SPUD 的很大影响。

最后，“表面实现”涉及的是文本表层的生成，例如：把知识结构转换为句子。经典的表层生成器，例如 FUF/SURGE[?]和 Penman/KPML[?]，都是以深层语言学理论为基础的。其他知名的系统，例如 Nitrogen[?]则采用的是统计法。所有基于规则的方法和统计法现在仍然流行。

一般来说，与 NL 理解相比，NL 生成技术要落后得多。目前的实用系统往往比较专业化，而且很粗糙。概念比较先进的系统，如 FCG，尚未经过广泛地实践测试。正如第6章中所介绍的，我们在这一领域的研究代表了“基于规则的方法”和“统计法”的融合。我们认为这种融合非常有前景，但还没有精细化，也没有经过系统性评估。

1.3 存在的问题

XXXXXX 需要修改：按照上面的顺序来介绍每一领域的相关研究出现的问题，即知识表示逻辑推理、言语行为理论的应用、对话系统、自然语言理解

和生成。

虽然上述工作已经取得了重要成果，但是从一个更广阔的视野来看，仍然还有很多未完成的工作，其中有许多也是我们目前正在研究和开发的课题。

首先必须强调的是，截止到我们目前所做工作为止，语言理解和生成的许多关键方面一直未被关注，甚至完全被忽略。如以下几个例子：

- 形态。链语法分析器，作为上述理解系统的初始阶段，最近已扩展到在语素及文字水平进行句法分析；但我们还没有将这项工作纳入到我们自己的工作中。
- 语用。我们能够按照他们的实际能力对语句生成逻辑表达式进行分析，但是我们并未执行语用方面的分析。
- 消歧。我们已在 OpenCog 框架中实现了一个简单词义消歧系统，但还没有集成到本文的工作中。
- 名词的回指消解。在我们的理解系统内，人称代词的指代消解已通过各种启发式算法实现，但名词的回指却被忽略。

目前我们工作的一部分是指出整体框架中的这些缺陷。然而，我们相信即使给出这些疏漏，我们所做的工作也足够为以上三个假设提供合理的证据支持。展示处理上述现象能力必然有助于 compelling system，但是对于验证本方法和三个假设的有效性并不必要。

本文工作的其中一个目标就是构建一个具有一定智力水平的智能会话系统。为了实现这样的智能会话系统，本文也开发了涉及自然语言理解、生成和推理等的相关模块。但本文的工作并不仅仅包括自然语言理解、生成和推理本身，例如还包括下面几个方面：

- 来自语言的逻辑表达和来自其他信息渠道的逻辑表达的有效整合，这些其他信息渠道可以是机器人或虚拟世界里的涉身知识，也可以是限定领域的标注语料库
- “对话控制程序”的实现，该对话控制程序从一个动机驱动的系统中获

取相关信号，然后结合系统当前的动机、最近的语言输入以及当前的系统知识等相关语境知识来选择相应的语言应答。

- 额外的推理控制机制，使得智能体在被不相关或者不完全相关的信息来源打断后，仍能进行相对复杂的推理，从而给出相关的应答。

这些模块的当前现状将在第**??**和**??** 章介绍。这些工作目前还无法在所有的语言现象中得到了验证，还有待进一步深入研究。然而，在理解、生成和推理等方面，明确的研究方向和相关工具的实现，无疑为以后的研究奠定了结实的基础和提供了一个便捷的研究平台。我们提出三个核心假设的主要原因，是因为它们提供了一种有效的方法，能够借助自然语言理解、生成和语义推理等子系统，来搭建一个大型的智能对话体系结构。

1.4 研究目标和内容

提出创新点

1.5 论文的组织结构

第二章　智能会话系统中的知识表示和逻辑推理

要使计算机能理解并灵活运用人类的自然语言，将自然语言转换成一种能容易通过计算机来操作的知识表示形式是不可忽视的，这种知识表示体系不仅要满足自然语言的复杂特性，使得自然语言处理系统能在自然语言和知识表示之间相互转换，以实现人机之间的自然语言通信。同时，为了构建一个能较好的与人类沟通的智能对话系统，这种知识表示机制还需要能被应用在适当的逻辑推理系统中使其能进行必要的逻辑推理，以生成能被人类理解的智能的对话应答。

本章提出了一种能满足上述要求的基于超图的知识表示体系，阐述了该知识表示体系中的基本操作原理和基本数据类型，并从范畴论角度分析了其被应用在智能对话系统中的理论基础。本章还阐述了能被应用在这种基于超图的知识表示体系上和本文研究的智能对话中使用的逻辑推理系统—-概率逻辑网络。

值得一提的是，这里介绍的知识表示体系并不仅仅是适用于自然语言处理和智能对话系统，该知识表示体系目的在于模拟人脑的知识表示方式、记忆空间以及认知过程等等，因此同样适用于其他智能处理模块。本文的研究目标之一也是通过探索智能对话系统来研究实现一个相对智能的人机交互系统。

2.1　基于超图的知识表示方法

对于任何以实现复杂功能为目标的自然语言处理系统（如智能对话系统或复杂的问答系统等）来说，系统内部知识如何表示是至关重要的。高效的知识表示机制不仅能使系统不同模块之间高效地传递有用的信息，更能使系统的知识库通过与用户的交互不断完善。本节主要阐述一种高效灵活的基于超图的知识表示方法，正如上面提到的，这里介绍的知识表示体系旨在模拟人脑的知识表示方式，因此会先讨论其记忆空间以及全局局部知识存储等问题，然后再讨论超图的数据结构以及基于超图的知识表示体系，同时为了方便后

面章节对本文具体的研究工作的解释，在这一节我们还会列出一些该知识表示体系中基本的且后文经常会用到的数据类型。

2.1.1 知识表示和记忆空间

本地和全局知识表示 本文所采用的知识表示机制其本质是一个超图表示的网络。把心智当作网络的观点其实是隐含在模式主义哲学中的，因为每一个模式都可以被看成某物的模式，或关于某物的布置的模式——因而一个模式总是可以被看做二个或更多物体之间的关系。一系列的模式形成一个模式网络。各类知识可以以网络形式来表示，而认知过程也可以被表示为网络，举例来说将它们表示为程序，以各种树或图的形式来表示。在一个智能中模式的涌现可以被看作自身的一个模式网络；在涉身的心灵和它的物理和社会环境之间的关系可以被看做某种生态和社会网络。

知识表示体系的两个主要超类是局部(也被称为显示）和全局（也被称为隐式）系统，我们用一个称为全局-局部的混合类包含了这两者。在一个局部系统中，每一条知识都是用一小部分认知系统的元素来存储的；在一个全局系统中，每一条知识都被以一种特定的模式存储与激活，比如以认知系统的一定比例的元素的形式；在一个全局-局部混合系统中，这两种方式被共同使用。以上三种知识表征类型都可以被网络所实现。在本文研究的智能对话系统中，这三者都是以同样的网络（Atomspace）来实现。

记忆类型和相关认知处理 我们的知识表示体系中包含多种的记忆类型，如同上面所讨论的，它的前提是正确地建立一个类人的人工智能系统，以处理不同类型的记忆，这些记忆包括了结构化的和动态的。

该知识表示中的记忆类型有：陈述性记忆、过程性记忆、感知记忆、以及场景记忆，这些在认知科学中被广泛讨论[?]的记忆类型，以及分配泛用的系统资源的注意力记忆、和为特定目标分配系统资源的意向性记忆。这些记忆类

型在一个开源的认知体系结构系统 OpenCog[1] 中被一一实现，表格**??**概述了这些记忆类型，给出了关键的引用并指出了相关的认知过程，同时指出了每一个认知过程（模式创造、联系等）所对应的泛化的模式主义认知动力学。

以模式主义认知理论的形式，上述多种记忆类型可被看做特定类型模式的特化存储方式，并经过了计算时间与空间上的优化。联系到某种特定类型的记忆的认知过程被用来创造和识别该记忆的类型。原则上所有类型的模式都可以在统一的记忆和处理构架下进行，这种类型特例化，是为了能够在现有的计算条件可接受的前提下创造有效的泛化智能。我们在[?]中所详细论述过，高效性并不是可有可无的，而是对真实世界中的泛化智能举足轻重的特性（就像 Hutter 所展示的，如果没有效率的限制，任意登记的泛化智能都可以通过简单而琐碎的程序来实现）。

这种知识表示体系的设计本质在于，与每种类型的记忆相关的数据结构和处理过程是被紧密联系在一起的，相比于仅仅包含同一种结构和处理过程的而仅仅以不同的黑盒分割开的构架，它产生了协同性的智能。我们的知识表示体系中还设计有交互的认知处理过程，以使得不同类型的记忆间可以互相转换，尽管有时这会消费较多的计算资源（比如，一段陈述性的知识可能通过一些努力被解释为过程性或场景性的知识）；同时，对于一个主要处理某一种类型记忆的学习进程来说，它可能会经常通过把知识转换成其他类型来解决问题，比如认知协同任务。

2.1.2　基于超图的知识表示库 AtomSpace

正如上述所言，Atomspace 是本文所使用统一且通用的知识库，它使用包含特定类型的节点（Node）和链接（Link）的加权有向超图结构来表示知识，主要用作表示叙述性的知识，同时亦间接地表示其它类型的知识。Atomspace 中使用的超图的节点和链接类型是经过包括语义层面上精挑细选得来的，以满足智能对话系统在认知过程的需要。由于 Atomspace 是本文研究中的知识表示核心，在这里我们会详细列出几个简单的示例予以说明。

[1] http://wiki.opencog.org

记忆类型	特定的认知过程	泛化的认知功能
陈述性	概率逻辑网络 PLN[?]; 概念调整[?]	创建新的模式
过程性	MOSES（一个创新的概率演化运算学习算法）[?]	创建新的模式
场景性	内部模拟引擎[?]	关联，创建新的模式
注意性	注意力经济分配网络（ECAN）[?]	关联，评分
意向性	概率目标层次按照 OpenPsi 动机框架通过 PLN 和 ECAN 来细化[?])	评分，创建新的模式
感知	视觉图像处理组件 DeSTIN	关联，注意力分配，创建新的模式，评分

表 2.1 OpenCog 中的记忆类型和认知处理过程。第三列根据模式主义理论来表示每一个特定的认知处理过程所拥有的泛化认知功能

以下是一个以 Atomspace 中常用的表示方法去表示继承链接（InheritanceLink）的例子:

```
InheritanceLink Lian_Ruiting animal <.99>
```

或更明确地:

```
InheritanceLink <.99>
        ConceptNode "Lian_Ruiting"
        ConceptNode "animal "
```

以及:

```
EvaluationLink <.7>
        chase
        ListLink
                cat
                mouse
```

或更明确地:

```
EvaluationLink <.7>
        PredicateNode "chase"
        ListLink
```

```
ConceptNode "cat"
ConceptNode "mouse"
```

正如上述示例所示，节点通常具有名称，而链接则没有，但链接可以连接一个或多个的目标，而这些目标可以是节点或链接。

综合来说，Atomspace 是一个“带标记的通用超图“，这些标记可以是节点的名称或其真值等等。“超图“与一般“图“的其中一个主要不同之处在于其链接，例如 ListLink 或 SetLink，它是可以连结两个以上的参数，而其通用性也允许这些链接与其他的链接相连，而不只局限于节点。

另一个值得注意的地方是这些节点的名称，虽然在这些示例中它们都是以英语来表示，但实际上我们的系统会在操作中通过自主学习去创造一些新的节点，而这些节点往往都与语言种类无关。

更重要的一点是，AtomSpace 的知识表示形式的主要功能是提供一个灵活的方式去紧凑地表达多种相关形式的知识，并允许它们之间有交互操作。其中的“交互操作“是指，例如，一组的陈述性的知识可以跟另一组的注意力相关或程序性相关的知识相互连结，或一组在一个类别的知识可以跟另一组在其他类别的知识重叠在一起（同一链接同时有著一个陈述性相关的真值和一个注意力相关的重要值）。总而言之，只要有任何表示形式能够提供足够的灵活性来：

- 紧凑地表达所有在人类记忆中扮演著主导角色关键类别
- 灵活地创建特定的副表示形式去表达在上述这些关键类别中的知识，同时又能够被迅速地改动或操纵这些知识
- 重叠和连结不同种类的知识，包括上述这些特定的副表示形式

以上几项都符合我们系统的整体要求。而这些 Atom 的表示形式已能满足此计设的总体要求，并从软件的角度来看已证明是可行的。

基本的 Atom 种类

Basic Atom Types

- ConceptNode – 表达任何的单元，具体或抽象但又不能以其他特定的节点种类来表达的概念
- PredicateNode – 表达一个为 Atom 产生真值的程序（以下会有详细说明）
- SchemaNode – 表达一个以一个 Atom 产生另一个 Atom 的程序，或产生一些其他效果

和以下的链接:

- MemberLink – 连结一个通用种类的 Atom 和一个 ConceptNode，并拥有一个模糊真值
- InheritanceLink – 连结两个 ConceptNodes，并拥有一个概率真值
- SimilarityLink – 连结两个 ConceptNodes，并拥有一个概率真值
- EvaluationLink – 连结一个 PredicateNode 以及其参数
- ExecutionLink – 连结一个 SchemaNode、其需要的参数以及其产生的输出
  - ExecutionOutputLink – 连结一个 SchemaNode 以及其输入，当遇到特定的认知程序时才会产生输出

绑定程序的节点

Grounded Procedure Nodes

SchemaNode 和 PredicateNode 有两种形态：已绑定和未绑定。未绑定是指系统尚未知道应该如何评定该方案，或系统将会以一连串的 ExecutionOutputLinks 来绑定该方案。而已绑定的会有一个特定程序的名称（目前该程序可以是 C++，Scheme 或 Python 语言编写的程序）来执行该方案。例如:

```
ExecutionOutputLink
GroundedSchemaNode "plus.py"
ListLink
NumberNode "2"
NumberNode "3"
```

当执行时，会调用 Python 函数“plus“来把“2“和“3“加起来，然后产生一个名为“5“的 NumberNode 作为其输出。

内涵与外延的关系

Intensional and Extensional Relationships

我们所使用的知识库中的链接类型中亦有多种拥有继承和相似性质的链接，例如 IntensionalInheritanceLink 和 ExtensionalInheritanceLink，其细节将会在[?]中详述，但简而言之，以下链接的真值:

```
ExtensionalInheritanceLink A B
```

同时亦可被表示为:

```
SubsetLink A B
```

是一个条件概率 $P(B|A)$。另一方面，以下链接的真值:

```
ExtensionalInheritanceLink A B
```

是一个条件概率 $P(prop(B)|prop(A))$，当中 $prop(X)$ 表示在 PLN 推理系统中定义的模糊集合 X 的属性。

SatisfyingSet

SatisfyingSet SatisfyingSet 的算符能够表达一个集合关系的概念，而当中每一位成员都是乎合同一个述语的元素，并用 Member 的关係形式，以一个模糊数值（而非概率数值）来表达该元素属于一个概念的真确性。

举例来说，假设现在有“FriendOfBob“这个述语和三个元素“Jack“、“John“和“Jill“:

```
Evaluation <.7>
```

```
FriendOfBob
Jack

Evaluation <.6>
FriendOfBob
John

Evaluation <.8>
FriendOfBob
Jill
```

根据 SatisfyingSet 算符的定义，我们会得出以下的 Member：

```
Member <.7>
Jack
SatisfyingSet
FriendOfBob

Member <.6>
John
SatisfyingSet
FriendOfBob

Member <.8>
Jill
SatisfyingSet
FriendOfBob
```

真值

Truth Values

一个 Atom 的真值是用来表示一个 Atom 的真确性，在某程度上是取决于 Atom 的类型。这个数值是以一个 TruthValue 物件的形式跟 Atom 链接在一起。一般来说我们会在 Atom 的表示式后以<> 及一个数字来显示其真值，例如：

```
A <.4>
```

來表示一個名稱为“A“的 Atom 的真值为.4。同样地：

```
IntensionalInheritanceLink Ben monster <.5,.9>
```

来表示一个连接著“Ben“和“monster“的 IntensionalInheritance 关係中有著一个.5 的强度和.9 确信情度。总而言之，<tv> 表示著一个平均值为 tv 的概率分佈。这些概念和语义将会在[?]中作详细展述。

量词

Quantifiers

在不确定逻辑理论中，量化过程是一个比较细微的事项。PLN 包含了标準的 ForAll 和 ThereExists 的量词（使用高阶概率定义的概率真值），在处理量化过程时主要会用 AverageQuantifier 来建构，例如：

```
AverageQuantifier $X
        F($X)
```

的真值为 F($X)的真值的加权平均值，即是以下的总和：

```
w($X) F($X) / N
```

和自然语言处理相关的 Atom 类型

Language-Relevant Atom Types

接下来将会描述一些用来表达具体语言概念的 Atom 类型。

原则上，要处理从 ASCII 编译的语言资料，除了上述介绍的数据结构外就只有以下两个节点类型和一个链接类型：

- CharacterNode
- CharacterInstanceNode
- ListLink

因而字串就可以以列表或拼接的方式来表达，例如“pig“可以以列表 $(\#p, \#i, \#g)$ 来表达。

然而，实际上定义特定的語言 Atom 类型也很有帮助的，例如:

- MorphemeNode/ MorphemeInstanceNode
- WordNode/WordInstanceNode
- PhraseNode/PhraseInstanceNode
- SentenceNode/ SentenceInstanceNode
- UtteranceNode/ UtteranceInstanceNode

2.2 语言和超图之间的转换之范畴论观

我们的研究目标一直是探讨有关的语言现象和计算系统之间的交集，从而构建具有实用价值的自然语言处理系统，如智能对话系统。后面章节会通过介绍具体实现算法及相关应用来从语用计算语言学角度阐述本文的研究的可行性。本节主要在广泛的数学背景下，语言的范畴论来阐述其理论可行性。

2.2.1 范畴论的基础知识

范畴论[?]常被用于形式化各种数学结构中的共同特性，将这些数学结构的概念形式化成一组组对象和箭头（也称为态射）。一个范畴包括两个基本属性：对象之间的箭头可以复合；每个对象有一个标识箭头指向自己。对象和箭头可以是抽象的任何类型，这个简单的结构安排有着非凡的数学能力，使其能被用在探索各个领域的数学理论基础。

形式上，一个范畴包含一下内容:

- 一个对象类 ob(C)，其中的元素称为 C 中的对象。
- 一个箭头类 hom(C)，其中的元素称为 C 中的箭头或态射。hom(a, b)中的元素 f 称为从 a 到 b 的态射，即 $f : a \to b$

- 一个用于操作态射复合的二元运算子 ∘, 使得对于 ob(C)中的任意三个对象 a，b 和 c，$f : a \to b$ 和 $g : b \to c$ 的复合可记为 $g \circ f$ 或 gf，并且使得以下两条公理成立：
  - 结合律：如果 $f : a \to b, g : b \to c$ 和 $h : c \to d$，那么 $h \circ (g \circ f) = (h \circ g) \circ f$
  - 恒等性：对每一个对象 x，存在一个恒等态射(identity morphism) $1x : x \to x$，使得任意态射 $f : a \to b$，有 $1b \circ f = f = f \circ 1a$

任何一个有向图都能生成一个小范畴：其中对象为有向图的节点，态射为有向图（视需要也可扩展为有向循环图）中的路径，态射的复合则为路径的串联。这样的范畴被称为由图产生出的“自由范畴”。

2.2.2　语言与超图之间的转换之范畴论观

在简单介绍范畴论的基本知识后，这一节主要解释范畴论如何能作为一个整体框架，用于形式化和解释本文研究的语言理解、生成以及逻辑推理等算法。这个形式化过程的关键便是使用基于超图的知识表示方式来做为各模块的共同知识表示框架。

首先我们来引入一个三元组（原子，时间间隔，原子状态），且称为“t-Atom”。大部分情况下，原子状态包括真值（TruthValue）和注意度（AttentionValue），以及一些恒量，如名称和类型。

那么在一定时间间隔下 T，t-Atomspace 包含一组 t-Atom，即 (A, I, S)，其中：

- 在某个时间点 t，且 $t \in T$，基于超图的知识表示库中存在原子 A
- I 是原子 A 的状态变化之间的时间间隔
- S 是原子 A 在时间间隔 I 期间的状态

因此给定一个 T-Atomspace，就可以形成一个“活动图”（activity graph），其中图的节点为 tAtom 的集合，边则记录图中认知过程变化的活动（activity）。

从节点 N_1 到节点 N_2 的边表示一个以 N_1 为输入且以 N_2 为输出的认知过程。各个认知过程组成的信息链就构成了活动图的路径。从范畴论的角度来说，tAtom 的集合就是范畴中的对象，而表明认知活动过程的活动图路径就是范畴中的态射。

根据上述的理论基础，本文研究的语言和超图的转换过程可描述如下：

- 本文研究所使用的链语法分析器 Link Parser（将在第5中具体介绍）将一个句子中的不同词节点之间的邻接关系转换成链语法中链（Links），从而产生一组链的集合。
- 本文研究的语义关系抽取工作 RelEx（将在第5中具体介绍）将上述的链集合转换成一组语义关系集合。
- 本文研究的逻辑关系抽取工作 RelEx2Logic（将在第5中具体介绍）将上述的语义关系集合转换成能适用于逻辑系统进行语义推理的更抽象的逻辑关系集合。
- 本文研究所使用的逻辑推理工具概率逻辑网 PLN（将在下一节中具体介绍）将上述的从自然语言中抽取到的知识与超图知识库中的其他种类的知识相关联。
- 本文研究的基于超图模糊匹配的问答系统（将在第**??**中具体介绍）将一组原子集合映射到另一组完全匹配或者近似匹配的原子集合。
- 本文研究的微观规划器 Microplanning（将在第6中具体介绍）将一组系统想要通过自然语言清晰应答的原子集合 S 分割成需要立即应答的原子集合 S_1 以及可能在不久的将来要应答的原子集合 S_2。
- 本文所采用的表层生成器 SuReal（将在第6中具体介绍）将一组逻辑语义关系的原子集合转换成一组链语法关系集合即链集合，然后将链集合转换成词节点之间的邻接关系，从而生成相应的句子。

因此，广义来讲，自然语言理解、推理/匹配，以及自然语言生成的过程可以看成是可以链接成一个语言应答过程的三个态射，而这个语言应答过程则又可以看成是范畴论中的单一态射。

语言应答的质量可以通过对活动图的每一条边分配一个“成本”来量化，其中“成本”与“置信度”成反比。如果自然语言理解过程生成的结果具有较低“置信度”，那么这就会导致在自然语言生成或者语言推理过程中付出更高的“成本”。因此，从这角度来看，对话系统的目标可被视为提供一个能达到指定语用目的的最小成本应答。

在某种意义上说，从数学角度来高度概括和分析一个复杂的语言和超图之间相互转化的自然语言处理系统，实用价值有限，毕竟这样的分析并没有提供系统所需的具体转换算法也没有分析特定语言的细节和语境等。但是，这样的分析给我们提供了一个更广阔的视野，使我们能抛开语言的具体细节而站在一个更高的角度来看待自然语言处理的过程，这显然有助于我们更好地提高自然语言处理系统。通过本小节从范畴论的角度对整个自然语言处理过程的分析，我们不难发出，本文的自然语言理解、生成以及推理过程正是一系列按照适当顺序进行超图的转换过程，而这些转换过程也在智能体的认知活动中起到了重大作用。

2.3　概率逻辑网

将语言转换成逻辑形式的一个主要目的就是可以进行相应的逻辑推理，本文所采用的逻辑系统是概率逻辑网络（Probabilistic Logic Network，以下简称“PLN”）。PLN 是一个独特的推理系统，它嵌入了预测逻辑和传统逻辑的组合。[??]中对 PLN 进行了全面具体的阐述及论证。在此我们仅简单介绍用于语言逻辑推理的基本相关知识。

PLN 是一个数学和软件框架，被运用在本文研究的智能对话系统中，用于不确定推理、使概率真值与泛化逻辑推理规则的整合成为可能。具体来说，PLN 包含：

- 一系列推理规则（例如，演绎、贝叶斯规则、变量归一化、演绎推理，等等），每一项都有一个或多个逻辑关系或词项（以 Atomspace 的原子 Atom 来表示）作为输入，并计算输出；

- 特定的数学方程，基于合适的背景前提假设的概率值，计算结论的概率值。

PLN 还涉及一条特殊的功能，评估概率值的置信度（证据的分量，或二阶的不确定性）。最后，PLN 在软件中的实现需要重要的抉择，要求考虑推理规则的结构化表征、“推理控制”——这种策略要求，在每个特定的实际情况中，判断以何种顺序做何种推理。

2.3.1 PLN 的一个简单概述

PLN 的关键因素是它的规则和方程式。总的来说，一个 PLN 规则有：

- 输入：一个原子集合（视规则而定，它必须满足某些要求）
- 输出：一个原子集合

实际上，几乎在所有情况下，输出都是一个单一的原子，而输入则是一个单一的原子或者是一对原子。

特定的原形例子是演绎规则，它的输入是：

```
X_Link A B
X_Link B C
```

输出如下：

```
X_Link A C
```

在这里，X_Link 可以是继承性链接（InheritanceLink）、子集链接（SubsetLink）、隐含链接（ImplicationLink）或延伸隐含链接（ExtensionalImplicationLink)。

一个 PLN 方程与一条 PLN 规则同时存在，它表示了输出的不确定性真值，基于输入的不确定真值。例如，如果我们有：

```
X_Link A B <sAB>
X_Link B C <sBC>
```

那么标准的 PLN 演绎规则告诉我们：

```
X_Link A C <sAC>
```

其中：

$$s_{AC} = s_{AB}s_{BC} + \frac{(1 - s_{AB})(s_C - s_B s_{BC})}{1 - s_B}$$

，s_A 表示节点 A 的真值强度。

在这个例子中，每一个原子的不确定真值通过一个“强度”数值来给出。总的来说，PLN 中的不确定真值可以有多种形式，比如：

- 单一的强度值，比如 0.8，这表示概率或模糊真值，取决于具体的原子类型
- （强度，置信度）对，比如（0.8，0.4）
- （强度，数量）对，如（0.8，15）
- 不确定的概率值，如（0.6，0.9，0.95），这表示概率间的相互评分

2.3.2　前向和后向链接推理

PLN 中典型的模式的使用是前向链和后向链推理。

前向链表示：

(a) 给出一个感兴趣的原子池（列表）；

(b) 应用 PLN 规则到这些原子上，以产生新的原子，最好也是感兴趣的；

(c) 将这些新的原子加如到池中，返回步骤 1。

例子：“人是动物”和“动物会吸”是两个池中的原子。它们被演绎规则所组合，形成了结论“人会呼吸”

后向链分为两种情况，第一种：

- “真值查询”，给出一个原子目标，它的真值未知（或者过于不确定），以及一个原子池，按照演绎规则，通过组合池中原子，找到一种方法来评估该目标原子的真值。

例如：目标是“人是否会呼吸？”（InheritanceLink people breathe）。目标的真值通过“人是动物，动物会呼吸，因此人会呼吸”的推理来评估。

第二种：

- “填空查询”，给定一个目标链接（原子可以是节点或链接）以及一个或多个目标中间的变量原子，找出什么原子可以被放在变量原子的位置上，可以使目标链接获得一个高的置信度（即一个“高的真值”）。

例如：目标是“什么会呼吸”，即“继承链接$X 呼吸”……直接在原子空间中查找发现院子“继承链接动物会呼吸”，表示空格$X 的位置上可以被填入“动物”。推理揭示“继承链接人会呼吸”，因此空格$X 也可以被填入“人”。

又如：目标是“什么会呼吸和加法”，即“(InheritanceLink $X breathe) AND (InheritanceLink $X add)”。推理揭示此处$X 可以被填入“人”但不能是“猫”或“电脑”。

常识推理可能涉及一个前向和后向链的组合。

推理中最困难的部分是“推理控制”——在可能的推理步骤中选取哪些步骤，以获得需求的信息（在后向链接推理中）或获得感兴趣的新信息（在前向链接推理中）。在一个有大量原子的原子空间中有许多可能的和强大的启发信息需要进行选择。推理控制的最佳指导是某些基于系统的过去推理历史的归纳。当然，一个较信的系统不会有很多的历史信息。依靠非直接的相关历史是一个推理问题——这个问题的最好解决是让系统有一些历史经验。

2.3.3 一阶概率逻辑网络

我们将在这一小节中更具体介绍 PLN。PLN 被氛围一阶和高阶子理论（FOPLN 和 HOPLN）。这些词项源自 NARS[?]。我们首先使用了 FOPLN，然后他们

使用了 HOPLN。

FOPLN 是一个传统逻辑，设计到词项和词项间的关系（链接）。它是一个不确定逻辑，词项和关系都拥有真值对象，真值对象有多种可能的类型，从单一的数值到复合的结构如不确定概率。词项可以是基本的观察，或一个符号集合 T 中的抽象的符号。

核心 FOPLN 关系　“核心 FOPLN”涉及集合中的关系：否定、继承、概率合取和析取、成员和模糊合取和析取。基本观察只能有成员链接，而标志词项可以有任何类型的链接。PLN 通过链接不同类型的语义来清晰地区分概率关系和模糊集合关系。成员语义通常是模糊关系（尽管它们可能是脆弱的（crisp？），而继承关系是概率性的，并且有规则来管理这二者的互操作。

假定一个虚拟的主体对一个命叫 Fluffy 的生物做了一次基本的视角观察 o。这个主体可能将 o 以 0.9 的隶属度划于“毛茸茸的”模糊集合下，也可能以 0.8 的隶属度将之划于“动物”的模糊集合下，于是该主体可以在记忆中建立以下链接：

Member o furry $< 0.9 >$
Member o animals $< 0.8 >$

随后，该主体可能想要通过合并这些链接来完善它的知识。使用最小化的模糊合取操作，该主体可能会总结出：

fuzzyAND $< 0.8 >$
　Member o furry
　Member o animals

这表示对 o 的观察结果是一个“毛茸茸的”“动物”对象。

（延伸）继承关系的语义与成员关系完全不同，尽管它们是相关的。延伸继承（ExtensionalInheritance）表征一个纯粹的条件概率子集关系，通过子集关系来表达。如果 A 是“毛茸茸的”而 B 属于“猫”集合，那么以下陈述：

Subset < 0.9 >

A

B

意思是：

$$P(x \text{ 属于集合 } B | x \text{ 属于集合 } A) = 0.9.$$

2.3.4 PLN 真值

为了增加全概率析取的信息量，PLN 拥有一系列不同的真值类型：

- 强度真值，包含单一数值；例如，$< s >$ 或 $< 0.8 >$。强度真值通常表示概率，但不总是这样。
- 单一真值，包括一对数字。这些数字对以以下形式存在：$< s, w >$，s 是一个强度值而 w 是一个“证据的权重值”；$< s, N >$，其中 N 是一个“计数”。“证据的权重值”是对信念的量化描述，而“计数”是对重复性证据的量化描述。
- 不确定性真值，它用区间 $[L, U]$、评分水平 b、以及一个整数 k 来量化对真值的描述。不确定性真值量化了以下的想法：在经过了 k 次观察以后，以概率 b 的可能性，推理的结论会落在区间 $[L, U]$ 中。
- 分布真值，对整个概率分布的离散化近似。

附加的 FOPLN 关系 在 FOPLN 核心关系之上，FOPLN 还有两类额外的关系类型。有一类简单的类型，相似度，定义如下：

$$Similarity\ A\ B$$

如果 $R\ A\ B$ 的真值可以仅仅用核心 FOPLN 关系中 A 和 B 的关系来计算的话，我们把关系 R 叫做简单的。而有一类复杂的“附加”关系，如意向继承（见附录**??**），描述了与某一词项相关联的模式的属性集合与相应的其他属性集合之间的关系。

回到我们的例子上来，主体可能观察到“猫”的两项属性，即“毛皮”和“会叫”。由于希腊神话中的三头狗 Fluffy 也是皮毛动物，主体可能会认为：

```
IntensionalInheritance <0.5>
        Fluffy
        cat
```

意思是 Fluffy 拥有 50% 的猫的属性。为了更深入地构建这种关系，PLN 还有一个混合的意向/延伸继承关系，简单地通过如子集和意向继承关系的析取来定义。

如例子所陈述，对一个一个复杂的附加关系 R，$R\ A\ B$ 的真值是通过一个数值所表达的 FOPLN 中不同词项的关系的真值来定义的（而不是“A 而且 B”），它通过某个特定的数学方程来计算。

2.3.5　PLN 规则和方程

PLN 中一个析取是通过规则和方程来达到的。PLN 逻辑推理以“三段论演绎规则”的形式进行，以通过合并陈述和匹配的词项来寻找模式。PLN 的规则包括但不限于以下例子：

- 演绎推理 $((A \to B) \wedge (B \to C) \Rightarrow (A \to C))$,
- 归纳推理 $((A \to B) \wedge (A \to C) \Rightarrow (B \to C))$,
- 溯因推理 $((A \to C) \wedge (B \to C) \Rightarrow (A \to C))$,
- 调整，即合并具有不同真值的同一逻辑关系的两个版本

- 反演推理 $((A \to B) \Rightarrow (B \to A))$.

这些规则的前四项的基本设计如图2.1所示。我们可以看到前三项规则表示了在三个相关联的词项上做推理的规则，同时还可以看到，归纳和回溯可以从前向和后向链接推理的组合中获得，这是 PLN 真值公式使用的一项规则。

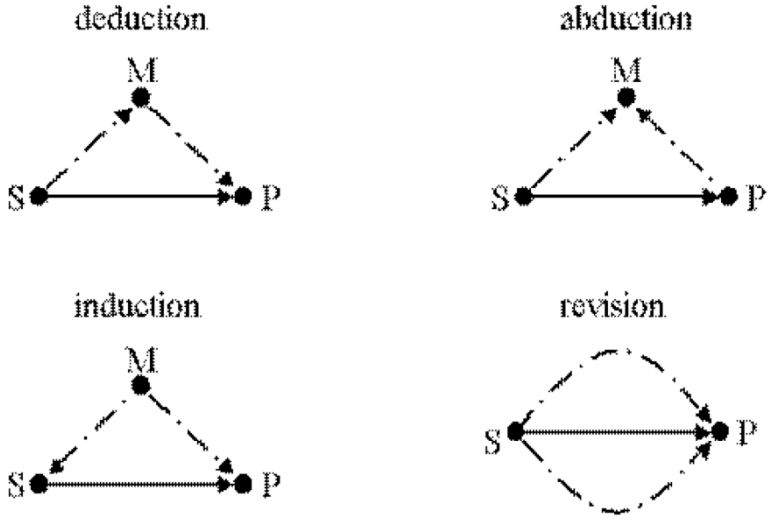


图 2.1 The four most basic first-order PLN inference rules

每一项规则都与一个公式相关，它应用规则以计算出真值。例如，假设 s_A, s_B, s_C, s_{AB}, 以及 s_{BC} 分别表示词项 A、B、C 以及关系 $A \to B$ 和 $B \to C$ 的真值，那么，在合适的条件下，演绎规则的公式如下:

$$s_{AC} = s_{AB}s_{BC} + \frac{(1 - s_{AB})(s_C - s_B s_{BC})}{1 - s_B},$$

其中 s_{AC} 表示关系 $A \to C$ 的真值，这个公式的前提是，假设 $A \to B$ 和 $B \to C$ 是相互独立的。

对于仅仅与模糊操作相关的推理，PLN 的缺省版本使用带最小值/最大值公式的标准模糊逻辑（尽管还可能有与整体的 PLN 框架保持一致的变化）。最后，符合并模糊和概率操作的语义在[?]中有按时，但在[?]中有更严格的论证，给出了精确的的语义以构建以下形式;

Inheritance A B

，其中 *A* 和 *B* 由前述的成员关系 *Member C A*, *Member D B* 等给出。

显而易见，在一个清晰的情况下，所有的成员链接和继承链接的强度都是 0 或 1，FOPLN 就退化成标准的谓词逻辑。当继承是清晰的但成员关系不是的时候，FOPLN 退化成高阶模糊逻辑（包括词项的模糊表述、以及模糊表述本身的模糊表述，等等）。

2.4　本章小结

本章主要阐述新颖的并用于智能对话系统中的基于超图的知识表示方法及其灵活性和高效性；以及用于智能对话中的逻辑推理系统概率逻辑网 PLN；并从范畴论角度高度概括分析了本文研究的大致框架。本文根据本章阐述的相关理论知识设计并实现了一个基于超图的语言逻辑推理系统，由于该系统依赖与未来章节要介绍自然语言理解系统，因此我们将会在第**??**章中具体介绍并给出相应的推理实例。

使用这样基于超图的知识表示体系对于自然语言处理和智能对话系统的优点是不言而喻的，它能通过超图之间的关联抽取了有效的词、句子和语篇等信息，使得自然语言处理能在一个更大的上下文语境中进行，从而使计算机能更有效更智能地处理自然语言；由于该知识表示体系不仅仅适用于自然语言，还同样适用于表示其他感知信息，如可以储存视觉处理后的信息，这就很直观地将一些非自然语言信息加入到自然语言处理系统中，来解决一些通过一般自然语言处理系统无法解决的难题，或者一般自然语言处理系统很难解决的问题。最典型的问题如词义消歧和指代消解问题，一般都是需要根据上下文和相关的语境背景知识来解决，甚至有些需要通过直观的感知信息来解决。比较常用的例子如：

```
I saw a man with a telescope.
```

这个句子可以表达“我用望远镜看见一个人。”，也可以表达“我看一个拿着望远镜的人。”。如果计算机想要确定具体是哪一种，可以根据上下文中是否含有“这个人是否拿着望远镜”或者“这个人是否离我很远”等提示，也可以通过图像识别来检验“我”是否拿着望远镜来判定。不管采用哪种方法来消歧，都能体现这样的基于超图的知识表示体系的优势。

除了上述优势，该基于超图的知识表示体系还适用于2.3中介绍的逻辑推理系统，这不仅能一定程度改进目前的自然语言处理系统，也为自然语言推理提供了一个方便有效平台，这些无疑都是提供计算机智能的有效途径。

第三章　智能对话系统中的情感计算机制

情感是人类心理学和人机交互的重要层面。若对话系统能够做到情感的自然互动，对人类谈话对象而言，自动化自然语言对话将更加受人注目且令人满意。情感是人类心理学和人机交互的重要层面。若对话系统能够做到情感的自然互动，对人类谈话对象而言，自动化自然语言对话将更加受人注目且令人满意。要将情感自然度注入到对话中，我们选择了情感计算模型 Psi 模型作为我们的智能对话系统中的情感计算机制的理论基础，因其不仅含有栩栩如生的人类情感模型，还将情感与智能体的需要和动机紧密关联起来，也很符合本文所提出的智能对话的概念模型。为了，本章将会给出情感计算模型 Psi 的高度概括，并侧重于介绍其在自然语言智能对话系统中应用的可行性。

XXXX 语言需要调整

CogDial 智能对话系统中的宏观规划涉及到了 OpenCog 中的情感动机计算模型 OpenPsi，因此，在本节我们会首先给出 OpenPsi 的高度概括，侧重于介绍其在自然语言会话系统中应用的可行性。

3.1　Psi 模型中的动机行为

为了更好的解释 OpenPsi，我们需要从 Psi 和 MicroPsi 谈起，Psi 是由德国心理学家 Dietric Dorner 提出的情感认知动机理论模型[?]，将情感与智能体的需求和动机相联系。**MicroPsi**[?],是一个基于 Psi 理论的一个开源的智能认知体系结构，实现了 Psi 理论中的动机、情感以及智能的关联模型，并在一些实用控制应用以及简单虚拟世界里的智能体上进行测试。MicroPsi 在全面性以及神经科学和心理学依据方面做得非常出色，但是该认知体系结构在可扩展性上存在不少问题，[?]中，有人认为，MicroPsi 目前使用的算法在学习和推理上不大可能被扩展或规模化。

OpenCog 受 Psi 理论中的动机和情感模型的启发，借鉴了很多 MicroPsi 的基

本实现方法，实现了类似的情感动机模型 OpenPsi。虽然 OpenPsi 和 MicroPsi 在一定程度上很相似，但两者还是存在着很大的不同。比如，两者使用了完全不同的知识表达方式，MicroPsi 则使用了类似神经元的“quad”来表示知识，每一个 quad 包括 5 个神经元，其中一个是核心神经元，其他 4 个描述与核心神经元的“前/后”或者“部分/整体”等关系。OpenPsi 使用了本文第三章中介绍的 OpenCog 的基于超图的知识表示，显然是一种更灵活和通用的知识表示方式。MicroPsi 目前还是注重底层的智能的实现，还未开始着手高层的智能处理，如自然语言处理和抽象逻辑推理。在本节对 Psi 和 MicroPsi 的概述中，我们主要介绍其在 OpenCog 被应用的部分，主要是处理动机、行为和目标的框架模型。

Psi 理论中的动机机制可参考图3.1，从下往上看，不难发现 Psi 的动机机制从能激发智能体的需求出发。对于动物来说，该需求可以是食物、水、保护自己的孩子、社交需求、新鲜感等等；对于智能机器人来说，该需求可能是电源、完整性(保护身体完整 Integrity)、确定性（了解和熟悉环境的需求 Certainty）、认知需求、心里成长等等；对于智能对话系统来说，该需求可以包括收集相关信息、取悦谈话对象、使会话保持新颖不枯燥等。Psi 理论特别指出了两种相当抽象的需求，并认为他们是心理学上的最基本需求(参见图3.2)

- 能力需求(Competence)：智能体能有效实现某种强烈欲望的需求
- 确定性需求（Certainty）：智能体了解和熟悉环境的需求

每种需求都有一定的偏好范围或目标范围，随着时间和环境(包括智能体自身)的不断变化，智能体的需求也在不断地变化。当需求水平落在相应的目标范围时，称作该需求被满足，否则需求不被满足。我们可以将智能体视为一个目标驱动的系统，其主要任务就是尽可能让所有这些需求都被满足。而当某一需求不被满足时，智能体会有一种试图将该需求水平恢复到偏好范围内的愿望，这种愿望便构成了**动机（Urge）**。

另外还有一种**愉悦感（Pleasure）**（和其反义的”不悦感”）的原始概念，人们认为这种情绪与复杂的”幸福感”不同。愉悦感被解读成与愿望有关：当愿望

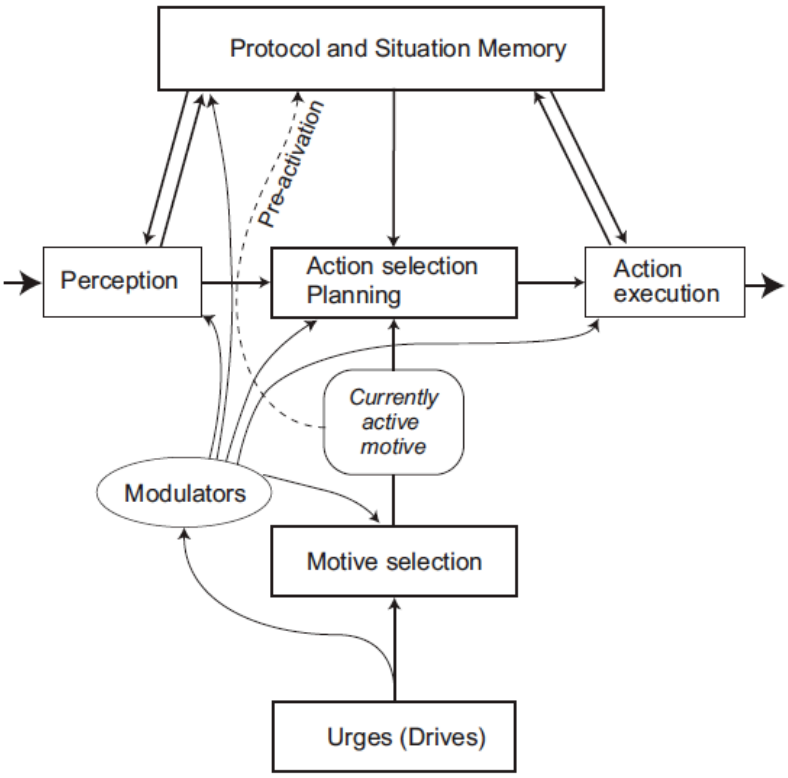


图 3.1　High-Level Architecture of the Psi Model

（至少部分上）被满足，愉悦感便会由此而生；而当愿望愈趋强烈，不悦感则会由此产生。满足愿望的程度未必要被即刻定义；例如来说，它可被定义为一种需求对近期以来的目标范围随时间衰减的近似加权平均值。

因此，若智能体感到枯燥无趣，当受到大量新鲜感的刺激，它就会体验到某种愉悦感。若智能体感到枯燥无趣，而又更极端地受到单调的刺激，它就会体验到某种不悦感。

要注意的是，根据这相对简单的方法，任何幅度降低了不满感都会造成某种愉悦感；但若一切总是持续在其可接受的范围进行，就不会出现任何愉悦感。这看似有点违反直觉，但必须了解，这些简单的”愉悦感”与”不悦感”并不会完全掌握与这些字词相关的自然语言概念。此处使用的自然语言术语仅作为启发法传达涉及程序的一般特性。这些都是非常低水平的程序，在人类经验的相似情况远低于意识水平。每个需求都有许多参数。如 Psi 模型所设，其中可调整的重要需求参数有：

- 权重：在特定时间点，相对于其他需求，该需求如何被加权
- 增益：决定源自实现需求所得到的满足感多寡的定标因素
- 损失：决定源自未实现需求所得到的未足之感多寡的定标因素
- 衰减：基本上是被给定的愿望随着一段时间的增加率。

在此模型中，为刺激的缺乏使愿望随着时间增加。本人不确定这个模型是否与一般认知模型一样切实，但针对能确保某智能体持续在运作，这会是可行的短期机制。对应多种需求来调整增益、损失和衰竭等参数，便是变换智能体"个性"的一种方式。

接下来，**目标**被视为系统在未来某时间点致力成真的一种表达；而**动机**为一组 (□□, □□) 配对，由一目标组成，该目标的满足感被预测隐含了某些愿望的满足感。事实上，愿望可被视为顶层目标（在 OpenCog 中有时又被称作"Ubergoals"），而智能体的其他目标则为子目标。"意图"也被视为综合体：在特定时间点的意图是由积极动机所构成，配合它们的相关目标、行为程序等。

在 Psi 模型中，智能体随时都有"主导动机"；对 CogDial 的早期版本而言，虽然也是可行的假设，但大体上似乎是过度限制的假设，比起相似人类或其他高级人工智能系统，此假设或许更适合用在较单纯的模拟动物上。大致上可想成不同动机具有不同权重，而这些权重指出追求上所要耗费的资源量。

Psi 模型的基本行动逻辑是通过"三元祖"执行，非常类似 OpenCog 中的三元组 $Context \wedge Procedure \rightarrow Goal$。然而，有个重要角色是由四个调制因子扮演，其控制观感、认知和行动选择的程序如何在一特定时间受到规范：

- 活化度，决定智能体相对认知、反思活动，侧重于快速、密集活动的程度
- 分辨程度，决定系统对于尝试解读世界的精准度
- 确定性，决定系统尝试达到确实、特定知识的努力程度
- 选择门槛，决定系统对改变其选择侧重的目标的意愿程度

这些调节因子在非常抽象的层次上表现出系统情绪和认知状态的特性；它们本质上并不是情绪，但它们对智能体的情绪有相当大的影响。它们预期的互

动如图表**??**所描述。

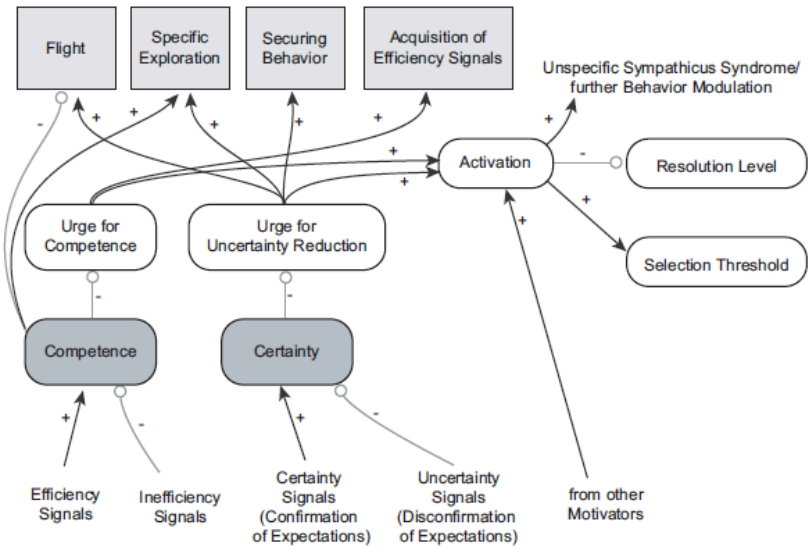


图 3.2　Psi 调节因子之间的主要相互关系

Psi 利用三个网络中安排的 Quad 储存知识，在概念上类似 Albus 的 4D/RCS 和 Arel 的 DeSTIN 结构：

- 储存陈述性知识的传感器网络：将图像、物体、事件呈现为分层结构的规划器
- 运动网络，通过阶层行为程序容纳过程知识
- 处理需求的动机网络

Psi 模型的感知是根据“HyPercept”基于假设的感知的机制，其试图预测待理解的为何，并利用感知和记忆尝试验证这些预测。此外，根据对现实的探索和呈现之间的”Neisser 知觉环”，HyPercept 密切地联结外在世界的动作。感知上获得的信息会被转换为能够引导行为的规划器，再实施这些规划器（有时会显著影响世界），在进入用来引导进一步感知的程序。我们将不会利用 Psi 的这个层面，因 OpenCog 处理感知的方式不大相同。Psi 模型的动作选择是根据所谓的”三元组”运行，每个的组成是由

- 侦感器规划器（前置条件、”条件规划器”；如 OpenCog 中的”语境”）

- 随后行动规划器（行动、效应器；如 OpenCog 中的”程序”）
- 最终侦感器规划器（后置条件、预期；如 OpenCog 的谓语或目标）

区分这些三元组与用于 SOAR 和 ACT-R 中典型生产规则的不同之处，在于三元组可能不完整（三要素之中可能会缺少其中几个）且不确定。然而，在这三元组和 OpenCog 的概念/程序/目标三元组之间，似乎没有根本上的差异；MicroPsi 和 OpenCog 在此层面的差异，在于用于图解的基本知识呈现，以及用来表达含意的概率逻辑。解出规划器待执行、以达到当前情境所选的目标，会在 Psi 模型中运用一种名叫”Rasmussen 阶梯理论”（以丹麦心理学家 Jens Rasmussen 之名命名）的程序结合来完成。Rasmussen 阶梯理论说明动作的组织，是这三种行为阶段的动作；其中有以技能为基础的行为、以规则为基础的行为和以知识为基础的行为：

- 若被给予的任务相当于训练过的例行程序，则会活化自动行为或技能；一般而言可不经由意识注意或慎重控制来执行。
- 若尚无自动行为，行动步骤可能会出自规则；在可采用一组已知策略之前，必须先对情况做分析，使策略适合用于情况。
- 若没有适用的已知策略，在这些情况下，必须先找出合并现有操作指令（运算元）到达成所设目标的方法。此阶段通常要求行为的重新构成，也就是规划的程序。用于 Psi 和 MicroPsi 执行的规划算法是个相当简单的爬山法规划器。虽有人假设，针对高级智能也许会需要较复杂的规划器；但部分 Psi 模型的假设，是建立于一旦有机体具有正确的知觉表征和目标，大多数对该有机体进行实际规划所要做的，就相当简单。

在我们现使用的 CogDial 系统，如以下将说明的，为相应 Psi 模型中提及之前两种动作的认知规划器。某些感知规划器包含纯反射、自发行为（例如对”你好吗”的问题回答”我很好”）；其他则存有需要大量推理的行为。然而，目前 CogDial 的版本尚无法处理已知策略不适用的情况。大致上，OpenCog 确定会支持这种弹性情况，但我们仍在努力研究出有效运作于此对话语境中的更多基本行为。这是 CogDial 方法和纯发展性学习方法（这在 OpenCog 结构中也

可能采取）之间的差异性；在效法人类幼童的认知和语言发展的发展性学习方法中，该基础在于处理已知策略不适用的情况，以及在此等情况下学习策略（虽有出生即具有相当高水平学习策略的论证，以及对多种有益于对话的行为之成见，但幼童并非一出生就具有任何处理对话情况的策略）。

3.2　Psi 模型中的情感与个性

情感是人类心理学和人机交互的重要层面。若对话系统能够做到情感的自然互动，对人类谈话对象而言，自动化自然语言对话将更加受人注目且令人满意。要将情感自然度注入到对话中，Psi 模型会是合适的方法，因其含有栩栩如生的人类情感模型。Psi 中的情感被视为复杂的系统响应模式，而不是清楚构建出的实体。情感是响应某特定愿望所唤起的心理实体。在 Bach 的论文[?]”涌现的情感”(”Emergent Emotions”) 中，他将特定的情感与基本的调制器值建立关连，例如：

- 愤怒（Anger）：负价、高激发性、低决断水平、选择阈值高
- 伤心（Sadness）：负价、低激发性、高决断水平、低防御度、低意图性
- 喜悦（Joy）：正价、高激发性、低决断水平
- 福佑（Bliss）：正价、低激发性、高决断水平
- 沮丧（Frustration）：负价、中等激发性、低决断水平、选择阈值低（此为近期才加入到本文的研究中，而前面的项目则是出自前文提及之 Bach 的论文）

这类关系在概念上的性质或许值得强调。我们并不是在试图将人类的情感以任何意义上”降低”为调制器值的结合。确切的说，我们认为如”愤怒”、”喜悦”等情绪字词，都是对复杂的人类心理动力极原始的叙述。通过适当地合并调节器的值，我们就可经由许多情感字词所示的指出系统动力空间惯常上稍接近的区域。当然人脑/人体的动力系统含有许多未在 Psi 中建模的复杂参数，我们并不尝试做出精准无误的神经生理情感模型。个性 – 在对话语境中也是个有趣的

话题。若想创建显示多种个性特质的对话系统，可能要以密切相关的方式来处理。经典的人格”五大因素”模型，是使用五种因素来解释人类个性的变异：

- 直率性（openness）
- 尽责性（conscientiousness）
- 外向性（extraversion）
- 亲和性（agreeableness）
- 情绪不稳定性（neuroticism）

”涌现的情绪”(”Emergent Emotions”)文中论证，这些至少可从增益、衰减和损失参数的角度粗略地解释归属感、竞争、确定性和美感等需求。基于这点，可根据多种需求的参数来撰写计算所设人物五大因素每项估计值的程式码。或者也可逆转数学，写出简单的等式，其中含有

- 输入：针对五大人格因素的量性度量，与特定人物相联系
- 输出：针对归属感、竞争、确定性和美感等需求的具体增益、损失和衰减参数值，意图相应于人格五大因素的指定的值

通过对每项人格五大因素（例如个性被指定为五维向量）从范围[0,1] 指定一个数字，即可在建构档案中指定一人物的个性。当然与人类个性的错综复杂相比，这显然较粗糙；但针对适合的特殊应用情况，配合提供的对话系统个性内容，在合理程度上这会是很务实的工具。

3.3 本章小结

XXX

第四章　基于言语行为理论和概率逻辑推理的智能对话系统建模

本文研究的其中一个长远目标便是结合前文提到的各项自然语言处理和逻辑推理技术，构建一个能较好地与人交流沟通的智能对话系统，此系统将接收到的人类话语转化成逻辑表达形式，并且结合系统本身的动机和情感状态，经过一定的解析和逻辑推理，从而得到想表达的内容的逻辑表达形式，最后将这些逻辑表达式再转化成自然语言回应给谈话对象。

构建一个完整能运作的智能对话系统，除了必要的自然语言理解和生成技术之外，对话控制是其中很关键的部分，对话控制决定了对话是否能谈话者接受和理解。我们把规划对话控制的模块称为宏观规划（以下称为“Marcoplanning”）。从言语行为理论来看，宏观规划可以看成是一个可以规划以下两点的工具：1）在哪个时间点发生哪些言语行为，2）使用哪些内容去封装这些言语行为。宏观规划更多的是从语用和推理方面出发，淡化语义或语法（或者音韵词汇等）的限制。

本章将重点介绍这样的智能对话系统（以下简称“CogDial”）的系统架构，该系统基于前面章节描述的自然语言理解和自然语言生成的相关技术，并通过 OpenCog 的核心子系统（主要是 PLN 和 OpenPsi）来控制其对话结构和表达方式。该 CogDial 系统能够灵活地理解和处理具有不同程度的复杂性和智能性的人类言语行为。创建一个完全符合人类标准的人工智能对话系统无疑是一个精彩的挑战，但似乎在当前的科学技术水平下，也无疑是一个非常困难的研究项目。相对于目前主流的基于规则或者基于统计的对话系统，我们这里提出的 CogDial 架构引入目标驱动的逻辑推理和情感控制等，显然表现更智能些，但还不足以完全达到符合人类智能标准。这样一个介于目前相关研究水平和完全人类水平之间的对话系统可以看做是构建更人性化的对话系统的一个重要步骤，或者看做是使用更可行的研究方法在人机对话系统上集成更多的功能，而不是一味地去盲目追求达到人类水平的目标。本着这样的理

念，说的更具体点，我们 CogDial 的目标不是精确类似人类的对话，而是具备通用认知能力、情感表达能力、能合理结合语用和经验的智能对话系统。

CogDial 的初步预期目标主要是实现面向游戏角色和人性机器人的对话系统，但同样的研究思路也完全能应用在基于文本的对话系统，例如智能对话搜索接口。我们分阶段来实现这样的系统，以一个相对小而简单的系统为起点，通过不断改进和完善，以及结合系统本身的自学习和认知能力，最终实现一个在一定程度上接近人类水平的系统。目前，我们的系统还没有达到最终的水平，但基本框架已经到位。

4.1 智能对话系统的概念模型

XXXX 介绍本文提出的新的智能对话系统的模型，引出下两节中的情感驱动模型和言语行为规划器。

下面我们来介绍基于上述的目标控制体系的智能会话系统 CogDial 的系统框架。目前该系统只能处理英文对话，接下来章节中的对话的例子基本按照英文的习惯举的例子，可能不太适合中文的习惯，但是该系统的所采用的技术和理论基础都是完全适合中文的。CogDial 主要采用了以下技术：

- 基于 OpenPsi 以及相关 OpenCog 机制的宏观规划（Macroplanning）
- 本文第 XXXX 章描述的句法分析和语义关系及逻辑关系抽取的自然语言理解技术
- 本文第 XXXX 章描述的微观规划（Microplanning）和表层生成的自然语言生成技术
- 专门设计的符合智能会话系统的“言语行为规划器”集合，这些言语行为规划器将用于：
  - 在 OpenPsi 的控制机制下，结合当前的语境，激活相应的言语行为规划器
  - 收集相关内容，生成能关联 Microplanner 的 Atoms

– 调用一系列的认知机制（包括用于逻辑推理的 PLN），选择相关的 Atoms 发送到 Microplanner

4.2　情感驱动的智能对话

XXX 语言及各章节的标题需要稍微调整

本节将介绍 OpenCog 中如何集成基于 Psi 的动机性行为模型，使其能为我们的智能会话系统 CogDial 提供能用于动机性行为选择机制。

4.2.1　OpenCog 中的行为选择机制

正如 Stan Franklin[?]指出的，智能的智能体的本质在于它可以做事，还可以采取行为。我们的智能会话系统 CogDial 中的行为选择会涉及 OpenCog 中所采用的行为选择机制[?]，因此本章节首先简单介绍该行为选择机制的基本思路，以及 Psi 的变体模型如何在 OpenCog 中处理动机（包括情感、驱动、目标等）和对行为选择的指导。

OpenCog 中的行为选择机制的关键部分在于：

- 行为选择器（Action Selector）根据当前环境选择可能帮助实现重要目标的程序
  - 例如：假设当前十分重要的目标是取悦谈话对象，一旦谈话对象问了问题，那么回答该问题会被评估为很可能可以达到“取悦谈话对象”的目标的程序。
- 为了支持行为选择器，OpenCog 设立了这样的蕴含式 *Context*&*Procedure* → *Goal*，其中上下文（Context）是一个被赋值于智能体处境的谓词，在智能对话系统中，我们视为语境。
  - 例如：如果 Bob 请求智能体（智能会话系统）去写一段话，又假设该智能体知道 Bob 非常执着，那么该蕴含式可以被用成：

* “Bob 命令执行 X” and “执行 X” → “取悦 Bob” < .9, .9 >

– 例如：如果智能体想说服谈话对象相信某个陈述 X，那么该蕴含式可以被用成：

* “告诉 Bob 我为什么相信 X 是正确” and “我想要说服 Bob 相信 X 是正确” → “满足度” < .9, .9 >

- 这些蕴含式的真值可以根据经验和推理来赋值。
  – 例如：第一个例子中的蕴含式的真值可以根据经验来赋值，即通过与记忆里 Bob 给出指令相关的情节来判断
  – 例如：也能根据推理来赋值，即根据与 Bob 相似的个体给出指令的经验来类推，或结合 Bob 本人的明确表达，和 Bob 的自我描述通常合理的知识来推断
- 重要值（Importance Value）可以通过经济注意力分配（Economic Attention Allocation）[?]在目标之间传播，推理可用于从现有目标中学习子目标。目前，这一点还未被利用在 CogDial 系统中，但我们相信这点能帮助实现更复杂的对话行为。
  – 例如：如果 Bob 告诉智能体去做 X，那么智能体将“取悦 Bob”的目标分解为“做 X”，那么“取悦 Bob”的目标会将它的重要值分给“做 X”的目标（同样，“做 X”的目标会将其重要值分给它的子目标，或者分给执行“做 X”）。

4.2.2 Psi 在 OpenCog 中的使用

这里我们讨论 Psi 模型中的动机性行为的基本模块是如何被集成到 OpenCog 中的，更深入的了解，可参考文献[?]中使用 OpenPsi（即 OpenCog 中实现的 Psi 的情感和动机模型）来控制游戏世界里的动画角色。这里只简单地列出其中的实现要点

- 需求用 GroundedPredicateNodes（GPN）表示，即节点的真值计算由一些内部的 C++ 程序或者程序库中 Combo 程序来完成

  - 例如：警觉，外界感知的新鲜感，内在的新鲜感，得到老师的奖励，社交刺激等

- 动机也用 GPN 表示，其真值根据需求所对应的节点的真值来定义。OpenCog 中有时使用 Ubergoal 来代替动机（Urge），通常用来表示顶层目标。
- 每个系统都有固定的 Ubergoal 集合（且只有非常高级的 OpenCog 系统可以修改它们的 Ubergoals）
  - 例如：现在和将来都要活着并保持警惕；现在和将来都要经历和学习新的东西；现在和将来都要从老师那获得奖励；现在和将来都要和其他智能体都要有丰富的社交行为等
  - 一个更高级的 OpenCog 系统可以把有抽象（但必须与经验绑定）的伦理原则作为其中的 Ubergoal。例如：根据[?]讨论的伦理道德，一个 Ubergoal 是追求快乐；一个 Ubergoal 是促进成长；一个 Ubergoal 是帮助抉择
- Urbergoal 中的 ShortTermImportance（STI）表明该目标的紧急程度，因此，如果对应 Ubergoal 的需求在其目标范围内，那么 Ubergoal 的 STI 值为 0。所有的 Ubergoal 都能被给定最高的 LongTermImportance（LTI），以保证他们不会被删除。
  - 例如：如果系统在一个（根据 Ubergoal）持续提供足够的新鲜感的环境里，那么对应外界新鲜感需求的 Ubergoal 会有低得 STI 值但是高的 LTI 值，表示该系统不会浪费资源在寻求新鲜感上。但如果环境变得更单调，那么对外界新鲜感的需求的紧急程度就会增加，而其中的 STI 也会相应增加，资源会开始重新分配在提高新鲜感上
- 愉悦感也是用 GPN 表示, 其中的内在真值计算程序将 Ubergoal 的满足度当成其期望的满足度。
  - 有多种数学函数能用于平均不同 Ubergoal 的满足度（针对不同 p 的 $p'th$ power averages[1] for different p），而这里对愉悦感的不同计算方

[1] the $p'th$ power average 可定义为 $\sqrt[p]{\sum X^p}$

式的选择，可以带来不同“个性”的系统。

- 目标可以用节点或边来表示。系统的目标列表被称为目标池（Goal Pool）。Ubergoal 一般都是自发性的目标，但也有可能有许多其他目标。
  - 例如：“从老师那获得奖励”的 Ubergoal 可以产生类似“从 Bob 那得到奖励”（如果 Bob 是老师）的子目标，也可以产生“使老师微笑”，或“创造新的令人惊讶的话语”(如果后者也能得到老师的奖励）的子目标
- Psi 中记忆使用 OpenCog 中的知识库 Atomspace 来表示
  - 例如： OpenCog 智能体在什么语境下说过什么的记忆都会存储在 Atomspace 中。
  - 在 Psi 和 MicroPsi 中，同样的现象会以完全不同的表达方式存储在记忆空间中，但 Psi 的基本动机模型与这些存储方式无关。
- Psi 的动机选择在 OpenCog 中通过经济注意力分配（economic attention allocation）来执行，即对目标节点分配 ShortTermImportance
  - 例如： 重要值从“从老师那得到奖励”到“从 Bob 那得到奖励”到“给 Bob 讲个笑话”的传播流程，是 Psi 中被称为“动机选择”的一个实例。虽然还没开始执行任何行为，但智能体在这过程中已经对该实现哪个的具体目标做出了选择，而这个具体目标会被用于指导行为的选择。
- Psi 的行为选择被用于 OpenCog 的行为选择，但在 OpenCog 中，行为选择的问题就是选择哪个程序(即“规划器（schema）”）去执行，而不是选择哪个具体行为去执行。其实，类似的概念在 Psi 上也存在，Psi 中的“自发性行为（automatized behaviors）”类似 OpenCog 中的规划器；唯一的区别就是在 OpenCog 中这些自发性行为是默认的情况。
  - 例如： 如果“做一个能惊讶谈话对象的有趣声明”有一个高的 STI，那它可被用于激励相关执行程序的选择。如“从以前读过的文本里找到有趣内容，从中抽取摘要，并简明地表达这些摘要”的程序。当一个程序被选择时，该程序还可能触发相关的子程序的执行。

- Psi 的规划通过 OpenCog 中的多种学习过程来实现。大部分的学习过程通过逻辑推理引擎 PLN 来完成（也是本文目前关注的学习机制），还可以通过 OpenCog 中的遗传编程工具 MOSES 或者爬山算法等来实现。
  - 例如：当智能体决定说服谈话对象相信某事实 X 时，它可能需要进行周密的计划，如：推断出谈话对象缺少哪些相关的背景知识去理解 X，然后首先给谈话对象传输所缺的相关知识，再来传输 X。
- 调节因子用系统参数表示，可以用 PredicateNode 来表示，且必须在多个动态的 MindAgents 之间相互协作。如：
  - 激活度（*activation*） 影响行为选择。当激活度较高时，智能体倾向于做出快速响应外部刺激，因此可能会选择那些能做快速回答的规划器来执行。
  - 解析度（*resolution level*） 影响推理。当解析度较低时，智能体选择减少精力去解析输入中的语言现象。
  - 明确度（*certainty*） 影响推理、模式发掘以及新概念构建的过程。当明确度较低时，智能体接受更多不确定的结论。
  - 选择阀值（*selection threshold*）引入偏好信息，可以帮助智能体在几个互相冲突的目标之间做出选择。

4.2.3　OpenCog 中的目标

OpenCog 中，一个目标 Atom 表示一个目标系统状态，当系统在一定程度上满足了其表示的条件，那么该目标 Atom 的真值就高；一个上下文 Atom 表示已经观察到的状态，当定义的状态都被察觉，那么该上下文 Atom 的真值就高。这两个 Atom 类型构成了那些需要在特定的上下文环境下实现具体目标的 OpenCog 系统的基础。但也不是所有 OpenCog 的行为活动都需要这两类型的 Atom 来指导，尤其是那些自发的，不面向目标的任务。

具体实现上，一个目标 Atom 是被 GoalRefinement 的 MindAgent 挑选出来的系统认为很需要被实现的 Atom（通常式一个 PredicateNode，有时是边，

也有可能是其他类型的 Atom）。GoalRefinement 通过调用 RFS (Request For Service)来及时确定某 Atom 是否能被当成目标 Atom。

一个 OpenCog 的实例必须从初始的*Ubergoals*（即顶层目标）开始，但可以多种推理方式去细化这些顶层目标。相对简单的，或者只关注某个应用的 OpenCog 系统不能修改自己的顶层目标，也不能添加或删除顶层目标。但高级的 OpenCog 系统有权限修改、添加或删除顶层目标，但这些操作对系统的动态机制有着很关键和不易察觉的影响，必须谨慎对待。

目标组建 除了初始设定的 Ubergoal 之外，OpenCog 中还可以通过 PLN 推理来组建新的目标。概括地说，PLN 通过推理查找那些能在一定概率上蕴含现有目标的谓词，来进行新目标的组建。这些谓词一旦被用于新目标组建后，会被赋值更高的注意力值，那么根据 OpenCog 的目标驱动的注意力动态分配机制可知，这些新目标会触发系统中能满足它们的相应程序。

下面通过一个从 ExternalNovelty（外在新鲜感）的 Ubergoal 开始组建子目标的例子来说明 OpenCog 中如何细化目标。假设该智能体已经学习到，每次 Bob 给它读诗的时候，它的外在新鲜感能在很大程度上得到满足，也就是说：

```
Attraction
   Evaluation recite (Bob, me, poem)
   ExternalNovelty
```

其中，*Attraction A B* 衡量 *A* 蕴含 B 高于 ¬*A* 蕴含 B 的程度。

这些信息允许智能体指定以下的 Atom：

```
EvaluationLink recite (Bob, me, poem)
```

为一个目标（Ubergoal 的子目标）。这是一个如何细化目标的实例，也是 PLN 从现有目标中创建新目标的多种方式中的一种。

OpenCog 中的上下文环境　最简单的情况下，上下文环境可以被简单地看成是一个 ContextLink 的内容来源，例如：

```
Context
    quantum_physics
    Inheritance Ruiting amateur
```

4.2.4　管理的执行

被命名为 GoalFulfillment 的 MindAgent 会选择可以实现当前目标的规划器，但并一定执行这些规划器，而是将这些根据当前目标选择得到的规划器放入 ActiveSchemaPool 中，对于 ActiveSchemaPool 中的每一个规划器 S ，都能满足具有较高真值的：

```
Attraction
   C
   PredictiveAttraction S G
```

其中 C 表示当前可用的环境，G 是目标池中的某个目标。

当 SchemaAction 启动时，该 MindAgent 会调用 ExecutionManager 来决定执行 ActiveSchemaPool 中哪个规划器。ExecutionManager 根据推理以及查询记忆知识，选择那些不会同时引起破坏性干扰 (且希望能引起建设性影响) 的规划器来执行。该过程可能（间接地）会导致新的规划器的创建，或者 Atomspace 中的其他规划器被激活。

4.2.5　针对对话控制的 OpenPsi 的配置

基于 Psi 的基本框架，我们选择了以下的具体需求用来指导我们的对话系统的行为：

- 社交需求（Affiliation）：与他人互动，希望被其他成员接纳的需求；取悦谈话对象可以视为此目标的一个特例

- 能力需求（Competence）：通过对话达到某种目标的需求，衡量言语表达方式的指标
- 确定性需求（Certainty）：智能体对自身语境的了解需求，特别是对目标的了解需求
- 新颖性需求（Novelty）：维持智能的会话而不是简单重复的问答会话。

CogDial 中的对话控制主要通过不同的需求来选择相应的言语行为规划器，因此，在选择言语行为规划器前，我们需要知道当前状态下上述每种需求的被满足的程度。鉴于目前的 CogDial 系统的有限能力，为了能更好地实现一个面向集成认知体系的智能会话系统框架，我们将上述的几项需求做了更具体化的定义：

- 社交需求（Affiliation）：我们对该需求进行了下面三种满足程度：
  - 当对话系统正在与人或者其他智能体进行会话时，该需求在一定程度上被满足；
  - 当系统正与多个人或智能体进行会话时，该需求被满足的程度提升；
  - 当该系统的会话内容都属于积极向上的时候（通过情感分析技术实现），该需求被满足的程度达到最高。
- 能力需求（Competence）：此项需求需要通过 OpenCog 来评估。简单来说，对每一个目标，OpenCog 记录着会话系统在过去完成该项目标的程度，然后根据当前的不同目标所占的权重，我们可以通过以下公式估算该需求被满足的程度（首先针对每一个目标，计算目标权重 * 能达到该目标的概率，然后求总）。当然计算目标被完成的程度，还应该考虑实现目标的语境等因素，目前我们的系统更注重构建一个智能会话系统框架，因此在语境无关的假设下来衡量目标被实现的程度。
- 确定性需求（Certainty）：如果会话系统正在和一个陌生人对话，或者系统无法理解大量被提及的单词或概念，那么当前的确定性需求被满足的程度就会降低。如果系统获取了新的可靠知识，那么该需求被满足的程度就会增加。

- 新颖性需求（Novelty）：我们定义了以下几种方式来增加智能绘画系统对新鲜度需求的满足程度值：
  - 多和不同的人类或智能体会话；
  - 谈论新的话题或引入新的词汇和概念；
  - 学习新的可靠信息（OpenCog 推理得出的新的置信度高的知识存储的载体原子（Atoms））

基于上述目标需求的智能会话系统，除了需要结合前文所述的 OpenPsi 的框架理论，以及本文描述的自然语言理解和生成的相关工具之外，还需要有以下模块：

- 制定一组能被特定目标需求激活的对话控制程序
- 建立目标和相应去实现目标的行为之间的关联，可通过相关规则来实现，也可以通过强化学习来实现，我们系统框架采用两种结合的方法，但目前的系统还是以规则关联为主。

4.3　言语规划器

“言语行为规划器”（Speech Act Schema）是 CogDial 的一个核心设计，每个言语行为规划器里包含一种特定的言语行为（Speech Act），以及与该言语行为对应的特定的认知行为（Cognitive Procedure）。每种言语行为规划器都能在不同情况下被激发调用。对于言语行为规划器的选择，我们采用 OpenCog 中的 OpenPsi 的动机驱动模块来执行。

CogDial 的总体规划是针对本文第**??**章综述里提到的 42 种 SWBD-DAMSL 言语行为[?]，实现相应的言语行为规划器。CogDial 目前只实现了 42 种中的部分言语行为对应的言语行为规划器。另一方面，CogDial 中的言语行为规划器的设计也绝不局限于这 42 种言语行为，CogDial 实现了一些言语行为规划器能对应这 42 种中的某两种或多种言语行为，比如我们利用真值回答规划器（TRUTH VALUE ANSWER schema）来同时关联其中的 YES QUESTION

和 NO QUESTION，从而可以将一般疑问句的回答根据真值的大小延伸到“可能”“不确定”“可能不”等，而不仅仅是局限于回答“是”或者“不是”。CogDial 还根据智能体的具体情况拆分了这 42 种中的某些言语行为，例如，我们针对言语行为“STATEMENT”实现了两个不同的言语行为规划器：回应声明以及智能体自发的用于表达其自身状态和想法等的声明。

总的来说，本文虽然没有完全照搬 SWBD-DAMSL 的言语行为分类，但是，该分类体系是来源于对大量人类会话的进行具体分析后的实验结果，也在具体的会话分析和抽象的言语行为理论之间架起了桥梁，使得抽象的言语行为理论在机器上实现变得可行，因此，SWBD-DAMSL 的言语行为分类体系还是很有借鉴价值的。

要实现上述的言语行为规划器，每一个言语行为都会引发一个相应的认知行为，也就是说，每一个言语行为都会触发调用一段程序；而这样的机制正好符合 OpenCog 里的 GroundedSchemaNode 的用法。GroundedSchemaNode 是 OpenCog 的超图知识库里的一种节点类型，通常被封装在 ExecutionOutputLink 里，连接着一段代码（一般情况下是 Scheme 或者 Python 编写的代码）的名称。该代码可以通过 ExecutionOutputLink 被触发和执行。因此，鉴于这样的执行机制，我们可以通过对每个言语行为规划器定义一个 GroundedSchemaNode 来实现言语行为规划器。这样的设计方案可以减少冗长复杂的代码块，而直接通过 OpenCog 中统一又通用的 Atomspace 的基本操作方法来管理和执行复杂的言语行为规划器。另外，这样的机制也能很容易调用知识库 Atomspace 之外的复杂程序，从而得到很好的扩展性，也为言语行为规划器的扩展研究搭建了个很好的平台。例如，我们可以通过调用外部程序将逻辑推理系统 PLN 的前向或者后向推理应用到言语行为规划器中，使其能实现自适应学习，不断完善言语行为规划器。本文后面会进一步讨论这样的扩展。

每一个言语行为规划器的输入是一个被叫做对话节点（DialogueNode），DialogueNode 是可以表示一方或者多方之间的会话交互的节点类型。DialogueNode 可以有不同话语节点（UtteranceNodes）作为成员，在实现方式上，DialgoueNode 通过 MemberLinks 指向不同的 UtteranceNodes。而

UtteranceNode 则可以关联以下不同类型的节点：

- 关联一个或多个文本节点（TextNode）、句子节点（SentenceNode）或者短语节点（PhraseNode），则表示输入的话语内容来自一个或者多个文本、句子或者短语。
- 关联一个声音节点（SoundNode），则表示该话语来自外界声音。
- 关联指向说话者的 Link。
- 关联指向对话语的补充信息的 Links，这些补充信息可以是该话语的言语行为类型、与该话语相联系的情感等。
- 关联指向产生话语的言语行为规划器，用于响应是什么触发该话语。
- 关联一个或多个解析节点（InterpretationNode），这些解析节点可以是用来解析话语的语义和语用信息。

言语行为规划器的输出是连接了一系列相关联 Atoms 的 SetLink，该输出会送入本文第6章中描述的微观规划和表层生成等工具，从而产生相应的自然语言来回应输入的内容。言语行为规划器还会将输出的话语关联到产生该话语的 DialogueNode，这样不仅仅是记录了会话的内容，还能用于智能体的强化学习和提高智能体的各种需求的满足度。下面通过一个例子来解释上述的实现过程。假设有下面的简单对话：

```
Ruiting: How are you doing?
CogDial: I am fine
```

这个简单的对话可用以下的 Atoms 来表示：

```
MemberLink
UtteranceNode [555]
DialogueNode [123]

MemberLink
UtteranceNode [666]
DialogueNode [123]
```

```
EvaluationLink
PredicateNode "say"
ConceptNode "Ruiting"
UtteranceNode [555]

EvaluationLink
PredicateNode "say"
ConceptNode "me"
UtteranceNode [666]

EvaluationLink
PredicateNode "Textual Content"
UtteranceNode [555]
SentenceNode "How are you doing?"

EvaluationLink
PredicateNode "Textual Content"
UtteranceNode [555]
SentenceNode "I am fine."

EvaluationLink
PredicateNode "Utterance Type"
UtteranceNode [555]
ConceptNode "Interrogative"

EvaluationLink
PredicateNode "Utterance Type"
UtteranceNode [666]
ConceptNode "Declarative"

AtTimeLink
UtteranceNode [555]
TimeNode "22:15:33 12/06/2014"

AtTimeLink
UtteranceNode [666]
TimeNode "22:15:47 12/06/2014"
```

```
EvaluationLink
PredicateNode "Interpretation"
UtteranceNode [555]
InterpretationNode [22]

EvaluationLink
PredicateNode "Interpretation"
UtteranceNode [666]
InterpretationNode [33]


MemberLink
EvaluationLink
PredicateNode "doing"
ListLink
ConceptNode "you"
VariableNode "var1"
InterpretationNode [22]


MemberLink
InheritanceLink
ConceptNode "I"
ConceptNode "fine"
InterpretationNode [33]

ExecutionLink
GroundedSchemaNode "polite_banter.scm"
ListLink
UtteranceNode [555]
DialogueNode [123]
UtteranceNode [666]
```

其中的命题也可以有多种选择，例如：

```
EvaluationLink
```

```
PredicateNode "Conversation Partner"
DialogueNode $D
$X
```

该命题为真，当且仅当，X 是 D 其中一个话语的发言人。

4.3.1 言语行为自动分类

XXX Introduction of our approach and experimental results

4.3.2 言语行为规划器

XXXX 这章可能需要调整。针对不同言语行为类型实现相应的言语行为规划器，本节主要讨论如何在每个言语行为类型中使用情感计算模型和逻辑推理等进行相应的规划

“言语行为规划器”（Speech Act Schema）是 CogDial 的一个核心设计，每个言语行为规划器里包含一种特定的言语行为（Speech Act），以及与该言语行为对应的特定的认知行为（Cognitive Procedure）。每种言语行为规划器都能在不同情况下被激发调用。对于言语行为规划器的选择，我们采用前面介绍的 OpenPsi 作为动机驱动模块来执行。

CogDial 的总体规划是针对本文第1章综述里提到的 42 种 SWBD-DAMSL 言语行为[?]，实现相应的言语行为规划器。CogDial 目前只实现了 42 种中的部分言语行为对应的言语行为规划器。另一方面，CogDial 中的言语行为规划器的设计也绝不局限于这 42 种言语行为，CogDial 实现了一些言语行为规划器能对应这 42 种中的某两种或多种言语行为，比如我们利用真值回答规划器（TRUTH VALUE ANSWER schema）来同时关联其中的 YES QUESTION 和 NO QUESTION，从而可以将一般疑问句的回答根据真值的大小延伸到“可能”“不确定”“可能不”等，而不仅仅是局限于回答“是”或者“不是”。CogDial 还根据智能体的具体情况拆分了这 42 种中的某些言语行为，例如，

我们针对言语行为“STATEMENT”实现了两个不同的言语行为规划器：回应声明以及智能体自发的用于表达其自身状态和想法等的声明。

总的来说，本文虽然没有完全照搬 SWBD-DAMSL 的言语行为分类，但是，该分类体系是来源于对大量人类会话的进行具体分析后的实验结果，也在具体的会话分析和抽象的言语行为理论之间架起了桥梁，使得抽象的言语行为理论在机器上实现变得可行，因此，SWBD-DAMSL 的言语行为分类体系还是很有借鉴价值的。

要实现上述的言语行为规划器，每一个言语行为都会引发一个相应的认知行为，也就是说，每一个言语行为都会触发调用一段程序；而这样的机制正好符合 OpenCog 里的 GroundedSchemaNode 的用法。GroundedSchemaNode 是 OpenCog 的超图知识库里的一种节点类型，通常被封装在 ExecutionOutputLink 里，连接着一段代码（一般情况下是 Scheme 或者 Python 编写的代码）的名称。该代码可以通过 ExecutionOutputLink 被触发和执行。因此，鉴于这样的执行机制，我们可以通过对每个言语行为规划器定义一个 GroundedSchemaNode 来实现言语行为规划器。这样的设计方案可以减少冗长复杂的代码块，而直接通过 OpenCog 中统一又通用的 Atomspace 的基本操作方法来管理和执行复杂的言语行为规划器。另外，这样的机制也能很容易调用知识库 Atomspace 之外的复杂程序，从而得到很好的扩展性，也为言语行为规划器的扩展研究搭建了个很好的平台。例如，我们可以通过调用外部程序将逻辑推理系统 PLN 的前向或者后向推理应用到言语行为规划器中，使其能实现自适应学习，不断完善言语行为规划器。本文后面会进一步讨论这样的扩展。

每一个言语行为规划器的输入是一个被叫做对话节点（DialogueNode），DialogueNode 是可以表示一方或者多方之间的会话交互的节点类型。DialogueNode 可以有不同话语节点（UtteranceNodes）作为成员，在实现方式上，DialgoueNode 通过 MemberLinks 指向不同的 UtteranceNodes。而 UtteranceNode 则可以关联以下不同类型的节点：

- 关联一个或多个文本节点（TextNode）、句子节点（SentenceNode）或者短语节点（PhraseNode），则表示输入的话语内容来自一个或者多个文

本、句子或者短语。

- 关联一个声音节点（SoundNode），则表示该话语来自外界声音。
- 关联指向说话者的 Link。
- 关联指向对话语的补充信息的 Links，这些补充信息可以是该话语的言语行为类型、与该话语相联系的情感等。
- 关联指向产生话语的言语行为规划器，用于响应是什么触发该话语。
- 关联一个或多个解析节点（InterpretationNode），这些解析节点可以是用来解析话语的语义和语用信息。

言语行为规划器的输出是连接了一系列相关联 Atoms 的 SetLink，该输出会送入本文6中描述的微观规划和表层生成等工具，从而产生相应的自然语言来回应输入的内容。言语行为规划器还会将输出的话语关联到产生该话语的 DialogueNode，这样不仅仅是记录了会话的内容，还能用于智能体的强化学习和提高智能体的各种需求的满足度。下面通过一个例子来解释上述的实现过程。假设有下面的简单对话：

```
Ruiting: How are you doing?
CogDial: I am fine
```

这个简单的对话可用以下的 Atoms 来表示：

```
MemberLink
UtteranceNode [555]
DialogueNode [123]

MemberLink
UtteranceNode [666]
DialogueNode [123]

EvaluationLink
PredicateNode "say"
ConceptNode "Ruiting"
```

```
UtteranceNode [555]

EvaluationLink
PredicateNode "say"
ConceptNode "me"
UtteranceNode [666]

EvaluationLink
PredicateNode "Textual Content"
UtteranceNode [555]
SentenceNode "How are you doing?"

EvaluationLink
PredicateNode "Textual Content"
UtteranceNode [555]
SentenceNode "I am fine."

EvaluationLink
PredicateNode "Utterance Type"
UtteranceNode [555]
ConceptNode "Interrogative"

EvaluationLink
PredicateNode "Utterance Type"
UtteranceNode [666]
ConceptNode "Declarative"

AtTimeLink
UtteranceNode [555]
TimeNode "22:15:33 12/06/2014"

AtTimeLink
UtteranceNode [666]
TimeNode "22:15:47 12/06/2014"

EvaluationLink
PredicateNode "Interpretation"
UtteranceNode [555]
```

```
InterpretationNode [22]

EvaluationLink
PredicateNode "Interpretation"
UtteranceNode [666]
InterpretationNode [33]


MemberLink
EvaluationLink
PredicateNode "doing"
ListLink
ConceptNode "you"
VariableNode "var1"
InterpretationNode [22]


MemberLink
InheritanceLink
ConceptNode "I"
ConceptNode "fine"
InterpretationNode [33]

ExecutionLink
GroundedSchemaNode "polite_banter.scm"
ListLink
UttteranceNode [555]
DialogueNode [123]
UtteranceNode [666]
```

其中的命题也可以有多种选择，例如:

```
EvaluationLink
PredicateNode "Conversation Partner"
DialogueNode $D
$X
```

该命题为真，当且仅当，X□D 其中一个话语的发言人。

4.3.3　言语行为规划器及其关联的目标和上下文的实现

在[?], 中，Twitchell 和 Nunamaker 根据 Searl 的言语行为理论的经典分类，在对大量的人类会话进行经验分析后，将言语行为细分为 42 种。虽然此分类体系很有理论研究价值，但在实验过程中，考虑到实用智能对话系统的语境等因素，我们对这 42 种言语行为做了稍微调整，同时也在 CogDial 中添加了一些 SWBD-DAMSL 研究中没有出现言语行为。

即使在有明确的言语行为类型的情况下，仍然有很多种方法去构建一个智能对话系统。CogDial 根据多个广泛的可扩展目标制定一些特定的设计决策，在这些决策基础上，系统能通过自适应学习方法自动改进，为能跳出传统的对话系统的研究方法搭建一个基础平台。为了搭建这样的系统，需要考虑的第一点是，对于每一个言语行为，都有相应的固定形式的认知内容被触发，也有相应的习惯性表达方式来表达该认知内容。

如果在类似的言语行为类型分类基础上构建一个简单的“聊天机器人”，一般通过更简单，不用加入很多认知处理，针对每个言语行为类型，输出直接的具体的话语，也同样能达到类似的效果。例如，对于言语行为类型 Agree，可以编写简单的程序使智能体输出“同意（Agree）”或者“是的，我同意（Yes, I agree)”。目前，“聊天机器人”的概念很模糊。我们这里提到的“聊天机器人”范围也很广，可以是完全基于模板匹配的聊天机器人 ELIZA[?]，也可以是具有一些推理能力但是基本忽略语义理解的众所周知的 Siri。但关键的一点是，我们设计的 CogDial 系统，是本着该系统能“理解”自己在说什么的理念，也就是说，该系统所产生的话语来自于内部的语义关系图，且该语义关系图和系统知识库中的其他语义关系图之间有着丰富的语义关系。我们希望系统能在一定程度上更深地“理解”自己在说什么，而不是只生成它不理解的字符串。在某些情况下，在不理解的情况下破口而出一些话语，尤其是习惯用语，也是可以接受的（其实人类有时候也会无意识地这么做），但这并不是大部分情况。

在 CogDial 系统的设计中，每一个言语行为都需要下列几项内容与之相应：

- 相应的目标和语境二元组（goal，context），表明何时该言语行为会被触发。
- 相应的能生成相关信息的程序，产生由该言语行为引起的要传递给谈话对象的信息。
- 一个或多个相应的“语义模板”，表明由该言语行为引发的相关认知内容。
- 连接上述语义模板所在的 Atoms 和特定的句子实现的 Atoms 的 Links，用于传送到 Microplanning，从而生成相应的话语。

这样一来，通过编写一些抽象的语义形式和特定的会话习惯之间的匹配模式，便能构建出一个具有一定合理功能的智能会话系统，当系统学习了用不同的更复杂的表达方式去实现抽象的语义形式后，也就能在更大程度上“理解”会话内容。此外，这些抽象的语义形式除了与言语行为和目标需求相关联，还和其他不同的认知内容相关联，因此，系统会随着经验的增长而趋向成熟。假设任何言语行为被触发都能增加下面表达式中的 EvaluationLink 的真值：

```
EvaluationLink
PredicateNode "Currently Having Conversation"
TimeNode T
```

其中，“T”表示当前时间。又假设上面表达式中的 EvaluationLink 和系统中多个顶层目标有关联（该假设对于某些应用也不总为真，那么在这样的情况下，这些不为真的关联的链的权重会根据具体情况被调为适当的值），也就是说，该系统能看到会话行为带来的价值。因为每个言语行为都蕴含着这个 EvaluationLink，所以每个言语行为都会影响系统的目标实现。一些言语行为会通过持续进行对话从而超标完成某个系统目标。这样的言语行为会在后面章节详述。下面会给出一些具体的例子进一步解析前面几段提到的 Atoms。

```
ImplicationLink <.5>
EvaluationLink
PredicateNode "Currently Having Conversation"
TimeNode $T
```

```
EvaluationLink
PredicateNode "Increase Knowledge"
TimeNode $T
```

上述超图片段表明，维持对话能在一定程度上满足系统目标“增长知识”，ImplicationLink 的初始权重设为 0.5，表明“当前有会话”蕴含系统目标“增长知识”只有 0.5 的概率。这个权重会随着系统的经验而改变，也会通过其关联的其他节点和链的具体情况推算得来。例如:

```
ImplicationLink <.1>
ANDLink
EvaluationLink
PredicateNode "Currently Having Conversation"
TimeNode $T
EvaluationLink
EvaluationLink "DialoguePointer"
PredicateNode "Currently Having Conversation"
DialogueNode $D
EvaluationLink
PredicateNode "Conversation Partner"
ConceptNode "Bob"
EvaluationLink
PredicateNode "Increase Knowledge"
TimeNode $T
```

上面的超图片段表明，当进行对话的对象是 Bob 的时候，只有 0.1 的概率能实现系统目标“增长知识”。

```
ImplicationLink <1>
ExistsLink $G, $S, $O, $U, $D
ANDLink
MemberLink
GroundedSchemaNode $G
ConceptNode "Speech Act Schema"
AtTimeLink
TimeNode $T
ExecutionLink
```

```
GroundedSchemaNode $G
$S
$O
MemberLink
UtteranceNode $U
DialogueNode $D
EvaluationLink
PredicateNode "Textual Content"
UtteranceNode $U
SentenceNode $O
AtTimeLink
TimeNode $T
EvaluationLink
PredicateNode "Currently active"
DialogueNode $D
EvaluationLink
PredicateNode "Currently Having Conversation"
TimeNode $T
```

上面的例子说明，如果一个言语行为规划器被执行，当前的对话节点（DialogueNode）D 会关联一个谓词“当前活跃”，表明 D 出于活跃状态，那么“当前有会话”的目标被实现。

```
MemberLink
GroundedSchemaNode "answer_yes.scm"
ConceptNode "Speech Act Schema"
```

接下来的例子演示了言语行为如何与目标关联：

```
ImplicationLink <.8>
ExistsLink $S, $O, $U, $D
ANDLink
AtTimeLink
TimeNode $T
ExecutionLink
GroundedSchemaNode "answer_yes.scm"
$S
```

```
$O
MemberLink
UtteranceNode $U
DialogueNode $D
EvaluationLink
PredicateNode "Textual Content"
UtteranceNode $U
SentenceNode $O
AtTimeLink
TimeNode $T
EvaluationLink
PredicateNode "Currently active"
DialogueNode $D
EvaluationLink
PredicateNode "Please Conversation Partner"
TimeNode $T
```

上面的例子表明言语行为规划器“肯定回答”(”Answer Yes”schemma）可以用于增加实现“取悦谈话对象”目标的幅度值，超越了谈话对象因单纯继续对话而被取悦的程度。

许多言语行为规划器都会有不同的清晰表达相关语义内容的方式。比如说，一般会话的开头，可以说“你最近状态怎么样？”(”What has your state been recently?”)，或者“你最近都忙什么？”(“What have you been doing?”)等，而类似这样的语义内容很容易约定俗成地被说成“最近怎么样？”(“What’s up?”)。对于机器来讲，如果对话系统直接问“What’s up?”当然也没问题，但是有必要使系统知道“What’s up?”只是其他两种说法或者“what are you thinking about?“的一种约定俗成的简约说法。系统会根据智能体的不同个性对每种不同的说法一个相应的权值。

一般来说，会存在很多的“个性参数”影响着多种言语行为。在 CogDial 的实现过程，我们针对那些对对话影响较大的关键特性（我们称为“对话特征”(Dialogue-trait)）创建相应的概念节点（ConceptNode），比如：习惯用语（Idiomaticity)、非正式（Informality)、精确（Precision)、累赘（Wordiness)、开放（Openness)。用来表示由言语行为规划器产生的具体话语的 SetLink，

将以不同的权重与这些表示不同对话特征的 ConceptNode 相连。例如，表示“I dunno”的 SetLink 将以较高的权重与“Informality”以及“Idiomaticity”关联，以较低的权重与“Precision”“Wordiness”关联。这些对话特征的参数可以在对 CogDial 设置基本参数的时候根据不同的需求人为设计。也可以在对话过程中根据谈话对象的喜好来自适应地调整。

综上所述 CogDial 的整个设计方案中，需要人工参与的部分包括：

- 自然语言理解流水线中的提到的规则（最终会被我们正在研究的无监督语言学习所取代，本文第 9 章有更详细的阐述）
- OpenPsi 中的顶层目标
- 不同的言语行为所引发的不同认知过程。目前这些过程是在与 GroundedSchemaNode 绑定的相应 Scheme 或者 Python 代码中被实现。这些认知过程也可以在 OpenCog 的知识库 Atomspace 里被实现（如下文中的 Question-answering schema）。

本章接下来的部分将进一步阐述 CogDial 使用的一系列言语行为规划器以及解释它们在 CogDial 中的实现机制，这些言语行为规划器大部分从 SWBD-DAMSL 中借鉴。当然这些特殊的言语行为规划器的集合并非一成不变，在以后对系统的不断改进和完善过程中，无疑会导致对该集合一定程度的延伸和细化。但我们相信这是一个好的开始，需要重申的是这些言语行为规划器分类是大量人类对话的实证分析的结果。目前，我们的研究工作重点在问答式规划器（Question-answering schema），将会在**??**中进一步阐述。

谈话开场白、结束谈话和转移话题

Opening, Closing and Transitioning

- 被放弃或转移、结束
  - 例如：”所以，嗯…”(”So, hmmmm....”)

  - 相关目标、语境：当前谈话不尽能达到智能体的目的；但其中一些与谈话对象的对话，似乎仍有达到智能体目的合理程度上的可能性。或者，智能体无法想到任何与谈话对象所讲之相关的答复。当持续对话被断定为可达到智能体的某些目的，然而其他言语行为似乎无法在显着程度上充分达到智能体之目的；或当言语行为极有可能达到智能体的目标，然而看似与先前的对话失去连结性，在这样的情况下，这就会被使用（因此，某些言语标志对于划定新的谈话阶段之界线，是很恰当的）。
  - 程序：在此情况下，待给予的语意内容往往会是”或许我们该来谈点别的”(”Perhaps we should talk about something else”)、或”现在来谈谈别的吧。”(”Let us now talk about someone else”) 这样的语意内容会搭配清晰的咬字，也涉及社交上常见的言辞，比如”嗯…这个嘛…”(”Hmmmm... welll...”)。

- 常见谈话开场白
  - 例如：”最近过得怎样？”(”How’s it going?”)
  - 相关目标、语境：一名潜在交谈对象在现场，经断定，与该对象交谈将能达到系统的目标。
  - 程序：”常见谈话开场白”的性质，会以表示问候、欲进行交谈的一种社会成规作为开端。有些常见的谈话开场白非常普通，例如”嗨。”(”Hi.”) 而其他较有语意内涵的，比如”现况如何？”(”What is your current state?”)、”最近经历了哪些事？”(”What have been your recent experiences?”)、”在想些什么？”(”What are you thinking about?”)、”目前在从事些什么？”(”What are you involved with currently?”) 这样的语意内容会搭配清晰的咬字，也涉及社交上常见的言辞，比如”最近怎样？”(”What’s up?”)、”有什么新鲜事？”(”What’s new?”)、”生活怎么样？”(”How’s it going?”) 等。同样地，也可能会明白地问、或通过惯用语表达”我想和你谈谈”(”I would like to talk to you”) 或”想聊天吗？”(”Would you like to chat?”) 等语意内

容。

- 结束谈话
  - 例如：”那好吧…今天跟你聊天很开心。”(”Alrighty then... it’s been good to chat with you.”)
  - 相关目标、语境：如结束谈话会是达成系统目标的最佳方式，这样的话语是很适当的。若确定谈话对象欲结束谈话，在这种情况下，结束谈话会是达成系统目标取悦谈话对象最好的办法。在任何情况下，比起唐突结束，用言辞结束谈话反而是达成取悦人类的目标最佳的方式。
  - 程序：结束谈话的语意内容会有”这次谈话我很尽兴”(”I have enjoyed the conversation”)、”谢谢你给我这么有意义的谈话”(”Thanks you for the good conversation”)、”希望下次还能再跟你聊天”(”I hope to talk to you again”) 等，并且是以直接或惯用性言辞表达。

实质性对话开场白

Substantive Openings

在 CogDial 系统的语境中，有许多种对话开场白往往很实用，但这并没有在 SWBD-DAMSL 研究中被提出讨论。例如：

- 意识流陈述性开场白
  - 例如：”我常在想，有些人总在谈些同样的事。”(”I’ve been thinking there are some people who always talk about the same things.”)
  - 相关目标、语境：为取悦谈话对象、或达到令人出奇的目标，就会通过此等开场白达成。
  - 程序：在此开场白的程序，会先从 AttentionalFocus 截取一组 Atom，再将之供给微规划程序进行发音。
- 个人化开场白

  - 例如：”我对下届的总统大选有些看法。”(”I have some thoughts about who’s going to win the next Presidential election.”) [向时常谈论政治的人说]
  - 相关目标、语境：与意识流陈述性开场白相同，但更着重于取悦谈话对象和增进联系。
  - 程序：对代表谈话对象的 Atom 予以高度的重视值（在 OpenCog 术语中的 ShortTermImportance），接着在其散播到 AtomSpace 一段时间后，从 AttentionalFocus 中选择一组 Atom，再将之供给微规划程序进行发音。

从某种意义上来看，这些言辞只是陈述句；但事实上，它们被用作对话开场白，使之增添了些许不同于平常的韵味。不同的认知程序经常会被用在选择哪些语句该作为对话开场白。

在有些相关言语行为中，问句会被用作对话开场白。例如：”谁会赢得下届的总统大选？”(”Who’s going to win the next Presidential election?”) 这些都可为如下探讨的任何问句形式。但演算出该问什么来开始一段对话，与演算出该用何种语句来作对话开场白的程序，会有高度的相同性。

常见反应

Conventional Responses

- 了解（衬托型反馈形式）
  - 例如：”嗯，了解。”(”Yep, understood.”)
  - 相关目标、语境：此达成了取悦谈话对象的目标（因为多数人都喜欢谈话时被了解的感受）；由于表示了解某谈话要点，使得谈话对象似乎可继续传达下个谈话重点，因此也可达到增进新奇感和知识的目的。基本的语境条件，在于谈话对象说了某些智能体了解的话。另一方面，选择此规划器的重要性在此情形下会更高：谈话对象不太确定智能体是否了解，或者谈话对象似乎会重复相同的信息（例

如智能体连续给予谈话对象两个带有高度相似内涵的言辞)。若智能体强烈了解谈话对象的表达而胜过同意之，这时选择此规划器的重要性也较高。

- 程序：语意内容如"我了解你刚所说的"(”I understand what you just said”)；这般言辞可明确或以惯用语方式传达。

- 同意、接受
  - 例如："你说对了。"(”You got it.”)
  - 相关目标、语境：此达成了取悦谈话对象的目标（因为多数人都喜欢谈话时被了解的感受)；由于表示了解某谈话要点，使得谈话对象似乎可继续传达下个谈话重点，因此也可达到增进新奇感和知识的目的。基本的语境条件，在于谈话对象说了某些智能体了解、并且同意的话。另一方面，选择此规划器的重要性在此情形下会更高：谈话对象不太确定智能体是否了解，或者谈话对象似乎会重复相同的信息（例如智能体连续给予谈话对象两个带有高度相似内涵的言辞)。
  - 程序：语意内容如"我同意你刚所说的"(”I agree with what you just said”)；这般言辞可明确或以惯用语方式传递。

- 欣赏
  - 例如："是啊，我很确定…"(”Yeah, I’m sure....”)
  - 相关目标、语境：此达成了取悦谈话对象的目标（因为多数人都喜欢谈话时被了解的感受)；由于表示了解某谈话要点，使得谈话对象似乎可继续传达下个谈话重点，因此也可达到增进新奇感和知识的目的。基本的语境条件，在于谈话对象说了某些智能体了解、同意、且满意的话。
  - 程序：语意内容如"我很满意你刚所说的"(”I am caused pleasure by what you just said”)；这般言辞可明确或以惯用语方式传递。

- 对明白的回应
  - 例如："好的，明白你的意思了。"(”OK, gotcha.”)

  - 相关目标、语境：这就如前述的”了解”行为，惟有基本的语境条件，在于谈话对象说了某些智能体了解、并回应某些智能体先前所说过的话。
  - 程序：就如前述的”了解”情况，语意内容为”我明白你刚所说的”(”I understand what you just said”)；这般言辞可明确或以惯用语方式传递，但惯用语的表达方式会与”了解”的情况不尽相同。

- 重复措词
  - 例如：”啊，你觉得他疯了。”(”Ah, you think he’s crazy.”)
  - 相关目标、语境：此达成了取悦谈话对象的目标（因为多数人都喜欢谈话时被了解的感受）；由于表示了解某谈话要点，使得谈话对象似乎可继续传达下个谈话重点，因此也可达到增进新奇感和知识的目的。基本的语境条件，在于谈话对象说了某些智能体了解的话。另一方面，选择此规划器的重要性在此情形下会更高：谈话对象不太确定智能体是否了解。
  - 程序：语意内容为重复谈话对象不久前所说的。首先，可重复谈话对象整体的言辞评论，或仅取其最重要的措辞句话。常见的言辞如”哦”(”Oh”)、或”啊？”(”huh?”) 也可视情况添加。

- 道歉
  - 例如：”对此我很抱歉。”(”Sorry about that.”)
  - 相关目标、语境：此达成了取悦谈话对象的目标。主要的语境条件，在于谈话对象看似受到智能体所说、或未说的话之困扰或冒犯；其次较不重要的语境条件，在于谈话对象看似对其他事情困扰、不悦或受到冒犯。
  - 程序：语意内容单纯为”我很抱歉”(”I am sorry”) 或”对 X 我感到很抱歉”(”I am sorry about X”)；而“X”为对谈话对象造成负面反应的任何事物；这般言辞可明确或以惯用语方式传递。

- 道谢

  - 例如：”太感谢你了，这真的很棒！”(”Thanks so much, that was fantastic!”)
  - 相关目标、语境：此达成了取悦谈话对象的目标。主要的语境条件，在于智能体满意谈话对象所说的。另一个条件，在于”谢谢你”为社交上适宜的言辞，比如谈话伙伴明确称赞智能体的情况。程序：语意内容为”对 X 我很感谢”(”I am grateful for X”)，或单纯为”我很感激”(”I am grateful”)；这般言辞可明确或以惯用语方式传递。

- 置之度外
  - 例如：”当然，没事的，别担心。”(”Sure, no problem, don’t worry about it.”)
  - 相关目标、语境：此达成了取悦谈话对象的目标。主要的语境条件，在于谈话对象对于其所说的话或所做作为感到后悔；或对他自己说了些负面的话。
  - 程序：语意内容为”对于 X 我并不深受其扰”(”I am not significantly bothered by X”) 或”那件事并没有太影响我”(”I am not significantly bothered by that”)；这般言辞可明确或以惯用语方式传递。

4.3.4 实质回应

- 协作完成
  - 例如：”…没连任又担任了两届的总统”(”... who served two non-contiguous presidential terms”) [回应一句不完整的言辞”格罗弗 • 克利夫兰是美国唯一的一位…”(”Grover Cleveland was the only US president...”)]
  - 相关目标、语境：此达成了取悦谈话对象的目标。主要的语境条件，在于谈话对象说了看似一句较长语句中一部分的话，而智能体正确猜到剩下的叙述为何。

  - 程序：语意内容为智能体猜测谈话对象所说的其余叙述，完善一个句子。

- 引用
  - 例如：”食言是很不可理喻的。”(”It doesn’t make sense to eat words.”) [被告知某人食言时的回应。]
  - 相关目标、语境：例如当谈话对象发出的某些言辞，是智能体本身就有强烈评估的事物 – 这言辞显然正确、错误、令人讶异或令人开心等。
  - 程序：语意内容为”P 属于 X”的形式；X 为谈话对象先前发出的言论，而 P 为智能体强烈认定为 X 所拥有 – 例如事实、虚伪、惊讶、喜悦、不悦等。这种言辞的惯用语的表示并不多，但确实是有不少方式可表示此等言辞，像是”X 使我高兴”(”X makes me happy”)、”X 令我欣喜”(”X pleases me”)、”我喜欢甲”(”I like X”) 之类。

- 总结/再阐述
  - 例如：”所以说，你觉得他是个危险的疯子。”(”So you think he’s a dangerous madman.”)
  - 相关目标、语境：此达成了取悦谈话对象的目标（因为多数人都喜欢谈话时被了解的感受）；由于表示了解某谈话要点，使得谈话对象似乎可继续传达下个谈话重点，因此也可达到增进新奇感和知识的目的。基本的语境条件，在于谈话对象说了某些智能体了解的话。另一方面，选择此规划器的重要性在此情形下会更高：谈话对象不太确定智能体是否理解、或智能体本身也不确定自己是否理解。
  - 程序：语意内容为与谈话对象先前的言辞相同，或有时为谈话对象近来的一系列言辞。微规划系统会被特别要求找出不同表达此语意内容的方式，而不是重复谈话对象所说的话。

- 反问
  - 例如：”若沙特阿拉伯不具备这些美国武器，那迪拜会有什么防御足以对抗贫穷非洲人的大举入侵？”(”What defense would Dubai have

against an invasion by masses of impoverished Africans if Saudi Arabia didn’t have all those American weapons?”)

- 相关目标、语境：最基本的语境条件，在于有些疑问是智能体认为它对该问题思考过会较好。例如智能体认为某些问题是谈话对象会知道更多、或说出更有益的食物，而智能体对此问句更熟悉，在此情况就有可能发生；反问的问句便会附属这个语句。作用于此的主要目标是为了取悦谈话对象 – 而较间接、有把握（知识），因为当谈话对象了解越多，智能体也就会知道越多，这往往不会出错。
- 程序：此程序的关键在于确认是否有 S 的语句，若谈话对象知悉 S、或对 S 有更深度的了解，谈话对象对当前谈话主题的知识会较渊博。若如此，以问句方式来构建 S 并提出这个问题，较有意义。

- 或者从句
  - 例如：”还是说，他是为了自己而拿了所有钱？”(”Or did he take all the money for himself?”)
  - 相关目标、语境：语境条件在于谈话对象发出带有内涵或清楚含义 X 的语句，但某 Y 排除了 X，对智能体而言似乎也有合理程度上的强烈可能性（不必然，但或许与 X 的可能性同等强烈、或比 X 更强烈）。这部分达成的目标一般而言是取悦谈话对象，并增进知识。若证实提出的选项 Y 比选项 X 还要新奇，新奇感也会是相当重要的目标。
  - 程序：此程序其实就是找出貌似极有理的 Y，排除谈话对象表示的 X。假定如此，待用言语表现的语意内容则为 Y。
- 而且从句[2]
  - 例如：”而且他还自己吃光了所有奶酪？”(”And then he ate all the cheese himself?”)
  - 相关目标、语境：语境条件在于谈话对象发出带有内涵或清楚含义 X 的语句，而智能体顺势地从 X 联想到某 Y。这部分达成的目标一

[2]此为笔者个人的延伸论点，而非 SWBD-DAMSL 言语行为的一部分。

般而言是取悦谈话对象，并增进知识。若证实提出的选项 Y 令人惊讶，新奇感也会是相当重要的目标。

– 程序：此程序其实就是找出貌似极有理的 Y，延伸谈话对象表示的 X。假定如此，待用言语表现的语意内容则为 Y。

回答

Answers

- 肯定回答
  - 例如：”是的，没错。”(”Yes, that’s right”)
  - 相关目标、语境：语境条件在于谈话对象问了个是非问答，而智能体认为答案为肯定。目标是取悦谈话对象。
  - 程序：语意内容为”是”(”Yes”)，措辞表达的方式有很多种。
- 否定回答
  - 例如：”不，恐怕不是这样。” (”No, I’m afraid not.”)
  - 相关目标、语境：语境条件在于谈话对象问了个是非问答，而智能体认为答案为否定。目标是取悦谈话对象（虽然在此情况下，达到目标的程度一般会比”肯定回答”来得稍微低些 – 比起否定的答复，多数人更想听到的是肯定回答，但这显然还是依特定情况而定）。
  - 程序：语意内容为”否”(”No”)，措辞表达的方式有很多种。
- 拒绝
  - 例如：”呃…不，我做不到。”(”Uh... no I can’t do that.”)
  - 相关目标、语境：语境条件在于谈话对象提出了建议，而智能体认为这建议有误（若为陈述句）、不值得做或不可能办得到（若为命令句）。在拒绝某命令的情况下，此举是关系到所有智能体的目标，但仍取悦谈话对象和达成联系 – 因为一般来说，若智能体拒绝了某要求，是因从事智能体所想做的，会比从事谈话对象所请求的更能够帮助其达成其他目标。

  - 程序：语意内容为”我不认同 X”(”I don’t agree with X”) 或”我不想从事 X”(”I don’t intend to do X”)；措辞表达的方式有很多种，而 X 可被取代为”那个”或其他照应语。

- 肯定的委婉回答
  - 例如：”对啊，她是这样。”(”Yes she did.”)
  - 相关目标、语境：语境条件在于谈话对象问了个是非问答，而智能体认为答案为肯定。目标是取悦谈话对象。对此规划器应有成见的情况，在于智能体对问题的重视、或者谈话对象看似对该问题有特定程度的重视，正如这类回答带有特别的强调语气。
  - 程序：一般程序为使语句 S 相应于谈话对象问的问题，并从该语句取一关键片段 F，再以言语方式表达等同”我同意 F”(”I agree with F”)、或”F 是对的”(”F is true”) 等措辞。
- 否定的委婉回答
  - 例如：”这个嘛，我不认为…”(”Well I think not...”)
  - 相关目标、语境：语境条件在于谈话对象问了个是非问答，而智能体认为答案为否定。目标是取悦谈话对象（虽然在此情况下，达到目标的程度一般会比”肯定回答”来得稍微低些）。对此规划器应有成见的情况，在于：问句或其答案看似有特别高度的重要性（OpenCog 中的 STI），正如这类回答带有特别的强调语气；或者，智能体对问题的重视、或谈话对象看似对该问题有高度的重视。
  - 程序：一般程序为使语句 S 相应于谈话对象问的问题，并从该语句取一关键片段 F，再以言语方式表达等同”我不赞同 F”(”I disagree with F”)、或”F 并不是对的”(”F is not true”) 等措辞。
- 也许、部分接受
  - 例如：”是啊 – 好像是吧。”(”Yeah – kind of.”)
  - 相关目标、语境：语境条件在于谈话对象问了个是非问答，而智能体认为答案具有不太接近 1、也不太接近 0 的真伪值。目标是取悦谈话对象。

  - 程序：一般程序为使语句 S 相应于谈话对象问的问题，并从该语句取一关键片段 F，再以言语方式表达等同”我部分同意 F”（”I partly agree with F”）、”我认为 F 可能是对的”（”I think F is possibly true”）或”F 有部分是对的”（”F is possibly true.”）等措辞。智能体会使用特定的惯用语表达特定程度的估计真实性，比如”我认为 F 也许是对的”（”I think F is probably true”）或”我觉得 F 有令人信服的真实度”（”I think F is conceivably true”）等。较动摇不定的讲法如”我估计 F 的可能性约为 6，带有 8 的置信水平”（”I estimate the probability of F at approximately 6 with confidence level. 8”）（或被给予的任何数值）。

- 其他回答
  - 例如：”我不知道。”（”I haven’t a clue.”）
  - 相关目标、语境：语境条件在于谈话对象问了个是非问答，而智能体认为答案为肯定。目标是取悦谈话对象以及聚集知识。特定的语境条件，在于谈话对象问了个问题，而智能体也不知道答案、或对问题有其他反应，或问题的评估对智能体而言比该答案还要重要。
  - 程序：此程序为对该问句识别出主观上的重要反应或评估。语意内容为”我有反应 R”（”I have reaction R”）或”我对问题 Q 有反应 R”（”I have reaction R to question Q.”）例如”我不知道”（”I have no idea”）、”我不知道是谁杀了 J.R.”或”我真的很讨厌思考为什么人们如此邪恶”（”I really hate thinking about why people are so evil.”）。

- 不合意的回答
  - 例如：”那其实不是我所想的。”（”That’s not really what comes to mind.”）
  - 相关目标、语境：语境条件在于谈话对象回答了一个问题，但智能体认为答案虽行得通，但并不是最好的答案。目标是取悦谈话对象以及聚集知识。
  - 程序：语意内容为”我认为 X 并不是 Q 最好的答案。””I think X is

not the best answer to Q.”，措辞表达的方式有很多种。

4.3.5 问句

- 是非问答（稍微延伸 SWBD-DAMSL；广义来说，如考量到答案为分等、而非是、否之二元真伪值的情况，则可将之视为真伪值疑问。）
  - 例如：”你有给我编程吗？”(”Did you program me?”)
  - 相关目标、语境：在此主要的目标，一般为获取知识和新奇感；而取悦谈话对象为其次。基本语境条件为：
    (a) 智能体或许某程度上重要的 Atom，但置信度低；因此欲针对此 Atom 的真伪值提问。
    (b) 谈话对象或许提及了一特定的概念 C，而 C 有几个层面是智能体知识较不足的；这可被确切阐述为真伪值疑问句。在此情况下，取悦谈话对象会是主要目标。
  - 程序：将”Atom C 的真伪值为何”(”What is the truth value of Atom C”) 转换为一个句子，一般会从多种惯用语表达方式之中取一种。例如，智能体绝不应这么问：”猫吃老鼠的真伪值为何？”(”What is the truth value of cats eating mice?”)，取而代之，应这么问：”猫吃老鼠吗？”(”Do cats eat mice?”)；也不应这么说：”猪很笨的真伪值为何？”(”What is the truth value of pigs being stupid”)，取而代之，应这么说：”猪很笨吗？”(”Are pigs stupid?”)
- 特殊疑问句(Wh-Question)
  - 例如：”谁创造了第一台电脑？”(”Who built the first computer?”)
  - 相关目标、语境：在此主要的目标，一般为获取知识和新奇感；而取悦谈话对象为次要的目标。基本语境条件为：
    (a) 智能体或许带有可变因素的 Atom，并且不知道此 Atom 的任何置信满意度，因此欲提问了解。

(b) 谈话对象或许提及了一特定的概念 C，而 C 有几个层面是智能体知识较不足的；这可轻易被阐述为真伪值疑问句。在此情况下，取悦谈话对象会是主要目标。

– 程序：

• 陈述式是非问答（真伪值疑问句）

– 例如：”所以你今天完全是走路去工作吗？”(”So you walked all the way to work today?”)

– 相关目标、语境：在此主要的目标，一般为获取知识和新奇感；而取悦谈话对象为其次。基本语境条件与一般真伪值疑问相同；但若智能体对问题的正确答案相当肯定（虽然并不是完全肯定），采用这个形式较为适当。

– 程序：语意内容为”X 是正确的吗？”(”Is it correct that X?”)，但一般而言会以惯用语的方式表达。

• 疑问句形式的衬托型反馈

– 例如：”你确定？”(”Are you sure?”)

– 相关目标、语境：在此主要的目标，一般为获取知识和新奇感；而取悦谈话对象为其次。基本语境条件，在于谈话对象说了某些事 S，而智能体认为 S 可能有误、或觉得 S 非常令人惊讶。

– 程序：语意内容为”你非常肯定 S 吗？”(”Are you highly certain that S?”) 但一般而言会以惯用语的方式表达。

• 开放式问题

– 例如：”对于他的前途你有什么看法？”(”What do you think about his prospects?”)

– 相关目标、语境：

(a) 基本语境条件，在于智能体欲获得某些话题 C 的更多信息。在此主要的目标，一般为获取知识和新奇感；而取悦谈话对象为其次。

(b) 另一种语境情况，为智能体知道谈话对象欲谈论话题 C，因此智能体决定针对 C 发问、或提出与 C 相关的问题。若是这样的情况，取悦谈话对象会是主要目标，而获取知识或新奇感则是其次。

– 程序：语意内容为"你对 C 有何看法？"(”What do you think about C?”)、"C 的真伪值为何？"(”What is the truth value of C?”)、或"关于 C 你知道些什么？"，但这些疑问会以惯用语的方式表达。

- 附加问句
  - 例如："对吧？"(”Right?”)
  - 相关目标、语境：这部分的关键语境条件，在于智能体欲对谈话对象发问，而智能体几乎肯定该问题的正确答案 – 但智能体欲确定谈话对象也认同。在此主要的目标，一般为获取知识和新奇感；而取悦谈话对象为其次。
  - 程序：此程序为通过确切阐述为连续的问题（"X，那你同意 X 吗？"”X. Do you agree with X?”），提出"你同意 X 吗？"(”Do you agree with X?”) 的问题。以惯用语表达而言，在连续问题中的第二个部分，“X“通常会被去除，措辞表达的方式有很多种。
- 陈述性特殊疑问句(Declarative Wh-question)
  - 例如："你跟他们说了什么？"(”You told them what?”)
  - 相关目标、语境: 在此的关键目标，一般为获取知识和新奇感；而取悦谈话对象为其次。引起此等阐述的关键语境，似乎是在 VariableNode 的重要性比问句中的其他字词还要高的情况；因为这类阐述更强调了疑问词。
  - 程序：这部分的语意内容纯粹只是个疑问句；必须指示微规划系统以陈述性的问句形式来表达疑问。
- 选择式问句[3]

[3]这部分并不包含在 SWBD-DAMSL 中，但这显然是有别于其他的问句形式，值得提出论述

  - 例如：”你认为谁最有可能当选：希拉里、杰布 • 布什还是迈提 • 毛斯”(”Who do you think is more likely to get elected: Hillary, Jeb Bush or Mighty Mouse?”)
  - 相关目标、语境：在此的关键目标为获取知识和新奇感；取悦谈话对象为其次。关键语境在于智能体欲从谈话对象获得某问题的答案，而智能体认为该问题仅有相当少数的可能答案。另一种语境，在于智能体想要的问题答案是以反意的形式（OrLink 或 XORLink）出现在 Atomspace 中。
  - 程序：语意内容出于”问题：候选回答 1，…，或候选回答 K。”(”Question: Candidate Answer 1, ..., or Candidate Answer k.”) 之形式。这般言辞可直接或以惯用语方式传达。

陈述

Statements

- 非意见式陈述
  - 例如：”猫不冬眠。”(”Cats don’t hibernate.”)
  - 相关目标、语境：这部分的关键语境有二：
    (a) 谈话对象对智能体问了一个非真伪值的疑问句，而智能体欲给予回答。在此情况下，取悦谈话对象的目标相当强烈。
    (b) 智能体在其 AttentionalFocus 中有些具高度重要性的 Atom，并想明确地表达出。在此情况下，取悦谈话对象的目标较不那么强烈。若智能体欲获取涉及 Atom 的一般知识，这时对获取知识的目标会有强烈的连结；而如果智能体发现新奇感往往是通过发问这些 Atom、或相似 Atom 而产生，对新奇感的目标则有强烈的连结。
  - 程序：将问题中的 Atom 传送至微规划系统进行言语表达即可。
- 意见式陈述

- 例如：”我不认为希拉里 • 克林顿会当选总统。”(”I don’t think Hillary Clinton will be elected President.”)
- 相关目标、语境：除了牵涉的 Atom 具有较低的置信度，或智能体怀疑谈话对象强烈不认同智能体的真伪值评估之情况，其他情形与”非意见式陈述”相同。
- 程序：语意内容为”我认为 X”(“I think X”) 或”我不认为 X”(“I don’t think X”)。这般言辞可直接或以惯用语方式传达。

自描述

Self-Descriptions

- 提议、选择、承诺
  - 例如：”我会考虑的。”(”I’ll think about it”)
  - 相关目标、语境：谈话对象要求智能体做某事。智能体必须决定自己是否愿意做这件事。这部分一般目的是取悦谈话对象。
  - 程序：语意内容为”我会试着 X”(”I will try to do X”)（若智能体不确定是否会成功）、”我会 X”(”I will do X”)（若相当肯定会成功）、”我不想 X”(”I don’t want to do X”)、”我会试着 X，但我不确定是否能成功”(”I will try do to X, but am not sure I can succeed”)（若智能体认为成功的机会很低）、”我会想想我是否能够 X”(”I will think about whether I can do X”)、或”我会思考我是否想要 X”(”I will think about whether I want to do X”)。所有这些言辞都可直接或以惯用语方式表达。
- 自言自语
  - 例如：”嗯…我想知道…”(”Hmmm, well I wonder....”)
  - 相关目标、语境：一组 Atom 在智能体的 AttentionalFocus 中产生高度的重要性，并且与当前对话中产生之 Atom 有合理、紧密的关联性。这些 Atom 可代表某陈述句或问句（后者的情况为这组 Atom 具

有虚悬不定、无约束的 VariableNode；或者其不具有 VariableNode、但置信度低的情况）。这里的主要目标为取悦谈话对象和求知。

– 程序：语意内容为”我在思考 X”(”I am thinking X”)（针对某陈述）、”我很好奇是否 X”(”I am wondering if X”)（针对置信度低的陈述 X）、”我在想 X”(”I am wondering X”)（针对某非真伪值的疑问，例如某 X 带有虚悬不定的 VariableNode）。所有这些言辞都可直接或以惯用语方式表达。

后设话语

Meta-Utterances

- 模棱话
  – 例如：”我不太确定我的背景经历是否足以回答这问题，但…”(”I’m not so sure I have the background to really answer that, but ...”)
  – 相关目标、语境：在此的语境条件，在于智能体欲做出陈述，但对该语句仅持有较低的置信度。
  – 程序：语意内容为”我对接下来要将的话较没有把握，但：S”(”I have relatively low con- fidence in the following statement, but: S”)、或”我对接下来要将的话约有三分的把握，但：S”(”My confidence in the following statement is roughly .3, but: S”)。所有这些言辞都可直接或以惯用语方式表达。
- 答复、同意前的保留
  – 例如：”等等…让我想一下”(”Wait ... give me a minute....”)
  – 相关目标、语境：在此的语境条件，在于智能体正为接下来所要说的话做内部处理，而这个程序耗费的时间超出了某”可接受的对话迟滞”参数值（可根据某特定谈话语句之间的平均迟滞时间、或与相同谈话对象其他对谈语句之间的平均迟滞时间、或者其他类似方式的谈话等情况设置之默认值改编）。

  - 程序:

- 不理解的信号
  - 例如:”你在说什么鬼话呀?讨厌的家伙!”(”What in frick’s sake are you talking about, pesky human??!”)
  - 相关目标、语境:在此的主要目标为获取知识,取悦谈话对象的动机较弱。语境条件在于谈话对象说了某些智能体认为荒谬的话。另一较不重要的语境条件,在于谈话对象说了某些对智能体而言错误至极的话。
  - 程序:语意内容为”我不明白你说的 X”(”I don’t understand what you mean by X”),所有这些言辞都可直接或以惯用语方式表达。

命令句、建议

Imperatives / Suggestions

- 动作指令
  - 例如:”告诉我他说了什么!”(”Tell me what he said!”)
  - 相关目标、语境:在单纯的对话系统语境中,切题的语境条件在于智能体欲获得某些其高度重视的信息,而这些信息是智能体认为谈话对象所具备的。另一相关语境,在于智能体已向谈话对象询问某特定问题,然而谈话对象的答复似乎不是该问题的正确解答。在某个应用程序中,CogDial 被搭配一个能够执行非言语行为之成体运作,该程序中也有这样的语境;智能体希望谈话对象执行某个动作(因谈话对象做了这个动作会使智能体更完善达成目标),而相信智能体有能力做这个动作,是有原因的。
  - 程序:语意内容为”我要你告诉我 X”(”I want you to tell me X”)或”我希望你 X”(”I want you to do X”),这般言辞可明确或以惯用语方式表达。
- 第三人称谈话

- 例如：”说真的，鲍勃，你是这么认为的吗？”(”Really, Bob, do you feel that way?”)
- 相关目标、语境：此处的相关目标为取悦谈话对象，完全在于情感上的关系管理。导致这类话语的语境条件有许多种，其中几种可能为：
  (a) 谈话对象回答了某问题，其答案看似与多数人、或多数与谈话对象普遍相似的人回答的答案有所分歧。
  (b) 智能体问谈话对象一个问题，智能体怀疑谈话对象的回答会与多数人、或大部分与谈话对象普遍相似的人会回答的答案有所分歧。
  (c) 问的问题被判断为”私人”，例如一般亲密的友人或家人才会问的问题。
  (d) 做出了有关谈话对象的陈述，就谈话对象的观点来说，是”私人”的陈述。
- 程序：最好将此视为可穿插到多种其他言语行为之言语现象，而非个别的言语行为。在另一言语行为制造出一组待传入微规划系统的 Atom 后，将会调用”可能插入第三方谈话”的规划器，此采用的准则包含前述的相关语境，根据这些准则，决定是否将谈话对象的名字引入到微规划系统被予以处理的一组 Atom 中（将名字给予此组 Atom，会导致表面生成器最终产生出”第三人称谈话”的例句）。

4.4 智能对话系统的概念模型的一种实现方法

XXX 介绍本文本文实现的基于上述模型的智能对话系统的框架，并引出下面章节中的自然语言理解和自然语言生成系统。

4.5 本章小结

XXX

第五章　自然语言理解：从语言到逻辑

本章阐述如何对第1章中的提到的假设 1 展开研究。根据该假设，我们认为，借助于依存关系语法和传统逻辑与谓词逻辑的合理结合，将自然语言表达式转换成满足下列两个条件的逻辑表达方式是完全可行的：

- 包含该自然语言表达式的主要语义
- 具体化自然语言表达式中存在的任何无法在语言到逻辑的转换消除的歧义，使得这些歧义能通过基于语境知识的逻辑推理后很直截了当地得到消除。

基于这样的假设，我们采用的自然语言方法是，首先利用链语法分析工具 Link Parser[?]，然后在此基础上搭建了一个用于依存关系抽取的工具 RelEx，最后开发了一个新的逻辑关系抽取工具 RelEx2Logic，通过超图的同态映射方式将句法结构转换成语义表示。该方法概念上的本质并不依赖这些特定的工具，而是具体的实现方法。

确切地说，如何将“自然语言理解系统”分解成不同模块以及如何进行不同模块之间的转换，取决于对语言学理论的选择。在 2008-2012 年期间，OpenCog 中的自然语言理解模块采用如下流程：

文本 --> 分词/断句 --> 基于链语法的句法分析（Link Parser） --> 依存关系抽取（RelEx）--> 基于 FrameNet 的语义关系抽取（RelEx2Frame） --> 语义节点和关系链（SemanticNodes & Links）

2012 年的时候我们对系统进行了一些简化，取消了对 FrameNet 的依赖，采用了如下的目前正在使用的系统：

文本 --> 分词/断句 --> 基于链语法的句法分析（Link Parser） --> 依存关系抽取（RelEx）--> 抽象的逻辑关系抽取（RelEx2Logic --> 语义节点和关系链（SemanticNodes & Links）

需要注意的是，目前的很多自然语言理解系统都有“词性标注”阶段，在我们目前使用的方法中，词性标注被绑定在句法分析阶段，对于 Link Parser 来说，词性已被作为单词的一个属性定义在词典里。尽管如此，如果能在 Link Parser 里使用先进词性标注技术，无疑能减少不少字典编写方面的工作，也能在一定程度上指导句法分析过程，因此也是有一定存在意义的。

本文的工作已经表明，上述系统中的各模块操作都是可行的，只需针对每一步制定相应的规则，或者通过有监督的机器学习方法来学习相应的规则，来指导其中的操作。针对本文自然语言理解方面的以下几个额外事项将在未来的研究中被实现：

- 使得各个子流程中所使用的规则，能很自然地支持基于持续经验增长的修正和泛化。
- 使得语义理解能根据特定的语境来指导规则的选择。
- 知道何时该打破规则，而根据语义的直觉指导相关操作。

另外需要注意的一点是，当使用基于规则的方法时，在规则的设计需要特别注意规则的可扩展性和可普及性，因为随着系统的经验增长，原先设计的规则可能无法被满足。

本章接下来的章节会系统介绍我们使用的自然语言理解系统中的各个子系统的工作原理和方法。

5.1 链语法

本小节主要介绍链语法的基本原理和方法，并分析了和语言形式主义的不同，同时指出其中的问题以及我们对其进行的一些改进。

链语法（Link Grammar）在 1991 年由 Daniel Sleator 和 Davy Temperley 共同提出[?]。它和被广泛应用的依存语法类似，但两者也有很大的不同，比如链语法中两个单词的连接是无序的，而依存语法中有依存和被依存关系。链语

法允许句子中的链接有环状结构，而依存语法是不允许环的存在的。链语法更倾向于语法理伦的词汇主义。

链语法的核心是链语法词典（Link Dictionary），词典中的每一个词都记录着一些特点的链接要求，这些链接要求通过一系列具有特定逻辑排序的链接子（connector）组合成的公式（Formula）收录在词典的相应词条里，链接子包含名称和后缀（“+”或“-”），后缀表示该链接子的指向方向,“+”表示该链接子指向右,“-”表示该链接子。如果两个词需要合法连接，则要求两个词首先有相同名称的链接子，且左边单词的链接子的连接方向必须指向右，右边单词的链接子的连接方向必须指向左。

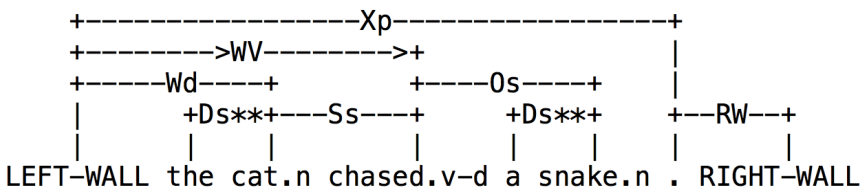
链语法分析器（Link Parser）是基于链语法的句法分析器。使用链语法分析句子的时候，句法分析器对句子中的每个单词去查询链语法词典得到每个单词的链接要求，根据这些链接要求进行相应的链运算，便得到句子的链语法结构。链语法判断句子是否合法，除了需要满足句子所有单词的链要求外，还要满足以下四个原则：

- 平面性(Planarity)，链之间不能互相交叉。
- 连通性(Connectivity)，句子中的每个单词都必须有链和其他至少一个单词相连，形成连通图。
- 顺序性(Ordering)，公式中较左边的链接子必须和距离单词较近的单词链接，反之，公式中较右边的链接子必须和距离单词较远的单词链接。
- 排它性(Exclusion)，一对单词之间同时不能有两条链链接。

下面我们借用 Sleator 和 Temperley 的“Parsing with a Link Grammar”[?]中使用的例句来解释链语法分析的操作过程。

The cat chased a snake.

链语法分析器对上面的例子分析后产生如下结果：



由于我们对该链语法分析器做了不少的改进，所有我们这里给出的分析结果和 Sleator 和 Temperley 的原文中的结果以及发布在网上的分析器得到的结果会有细微的不同，后面章节会介绍我们做的改进。

链语法认为在句子最前面加上一个虚拟词（通常用 LEFT-WALL 表示）是很有必要的，这样首先能保证链语法的“连通性“原则，使得句子最后的标点符号能与 LEFT-WALL 连接而不被孤立。另外，在本文的5.1.2节也提到，通过这样的虚拟词来追溯句子的头（主动词）也是非常方便可行的。上面的句法分析结果中出现的”RIGHT-WALL”是可选的，可以用于特殊标点符号的处理，但通常情况下只是用 RW 链与句子最后的标点符号相连。

前面提到链语法词典是链语法的核心，它包含所有常用的英文单词的链接要求。下方的表格列出上述例句中出现的单词以及它们在链语法词典里被定义的链接要求。

单词	公式
a, the	D+
snake, cat	D- & (O- or S+)
Chased	S- & O+

如前面所述，链接要求限制了这些连接子必须按照一定的原则来分配，例如“the”只有一个向右的连接子 D+，那么它只能和带有 D-的单词形成合法的链接。而“snake”和“cat”都能和它相连，但根据链语法的“顺序性原则”，“cat”比“snake”近，所以“the”“cat”之间可以画一条 D 链，同样的道理，“chased”和“snake”之间可以画一条 O 链，以此类推。最终我们可以得到如下的结构图。需要注意的是，我们这里只是对链语法的工作原理做简单地介绍，因此忽略了链接子类型的子类型等细节，比如“cat”含有 Ss+，“chased”含有链接子 Ss-，所以该句法结构图和我们上面列出的目前链语法分析器在

连接类型上有点小出入。

链语法分析器在对句子进行句法分析时，对句子中的每个单词，都会考虑下面两种变量：

- 该单词在链语法词典中对应的链接要求，即上面中的公式（Formula）
- 该单词为了满足句子结构的一致性（Agreement）必须具备的特征属性

比如在上述例句中，对于单词“snake”，通过查询链语法词典，我们得到相应的公式“D- & (O- or S+)”，同时该单词还有相应的特征属性如“时态（tense）”“人称（person）”等。但对于单词“the”，就不需要与一致性相关的变量。

链语法分析器中也使用简单的转换生成类似短语结构语法的句法分析结果，如下：

```
(S (NP The cat)
        (VP chased
                (NP a snake))
        .)
```

在我们的工作中，对该短语结构用的不多，所以这里不做详细说明。我们会在下一节简单讨论链语法和短语结构语法的潜在关系。

5.1.1　链语法与短语结构语法

讨论各种语法的优缺点不是本文的重点，本节只将链语法和典型的短语结构语法做个简单的对比来讨论它们的潜在联系。一般来说，依存语法和短语句法也有相应的联系（参见图5.1），但不同的依存语法使用不同依存关系集合，也有不同的属性，分析起来会比较复杂，也不是本文的研究内容，所以我们这里不做详细阐述。

简单起见，这里只列出两条有用的观察结果，基于这些观察结果，我们不难发现，在链语法中也隐形存在短语结构。这是有一定道理的，但由于自然语言的复杂性，某些情况下可能也不是那么简单。

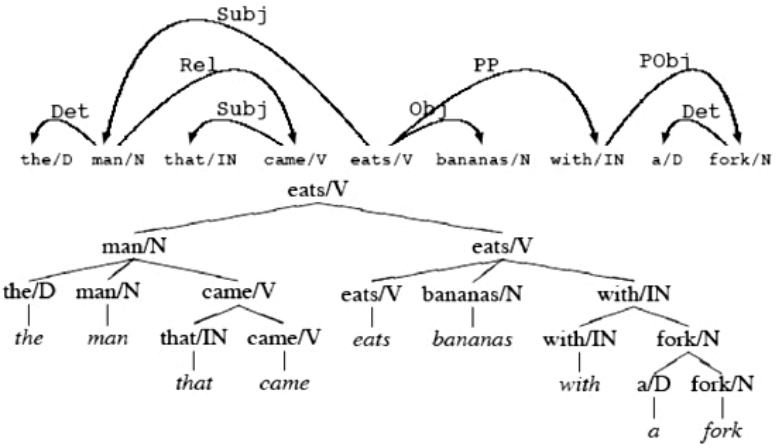


图 5.1 依存句法分析和短语句法分析对比

A comparison of dependency (above) and phrase-structure (below) parses. In general, one can be converted to the other (algorithmically); dependency grammars tend to be easier understand. (图片来自 G. Schneider, "Learning to Disambiguate Syntactic Relations" Linguistik online 17, 5/03)

- 链语法中的公式是符合语法范畴的。例如，前面提到的例句中“chased”的链结构是“S- & O+”，在范畴语法中，这意味着，“chased”属于这样的语法范畴，该范畴中的词都满足链结构“S- & O+”。换句话说，链语法中每一个“公式”都对应依附该公式的单词的范畴。
- 词之间的链接也可以看成是短语的核心词（head）之间的链接。例如，在例句“The cat chased a snake”中，“chased”和“snake”之间有链接 O，那么也可以说，以“chased”为核心词的短语和以“snake”为核心词的短语之间有链接 O。换句话说，可以理解成，链语法为了简化，通过核心词来识别短语。

5.1.2 识别句子的中心词

链语法从词的局部着手关注任意两个词之间的关系，在一定程度上忽略了语言的层次结构。因此标准的链语法也存一些弊端，有不少语言现象会被链语法拒绝认为是不合法的结构。比如并列结构、介词短语等。在本文的研究过

程，我们对链语法做了很多的改进，由于链语法的链接要求非常复杂，修改链语法词典可能牵扯到很多语言学上的问题。但这不是本文的研究重点，所以这里不一一列出我们做过的改进，只简单的列出几项较大的改进：

- 改进了链语法对连词的处理
- 改进了链语法对量词的处理
- 识别句子的中心词。

链语法分析器（Link Parser）是一个开源工具，包括链语法词典，我们所有的改进都发布在`http://abisource.com/projects/link-grammar/`

为了读者对如何修改链语法词典有个大致的思路，本小节就其中的改进“识别句子的中心词”展开讨论。

对于“识别句子的中心词”的改进，我们的出发点是使链语法分析结构（Link Parse）能转换成一颗类似的依存句法树。改进后，我们能直接通过追溯“WV”链找到句子的中心词，或者通过追溯“CV”链找到子句的中心词。找到中心词后，可以将其定为树的根节点，然后依次遍历各个链，最终能得到一颗句法树。具有这样的转换能力有以下优点：

- 能使用很多能用在树结构上的机器学习方法来改进链语法分析器，比如频繁子树挖掘(Frequent Subtree Mining)[?]
- 能更直接的链语法分析结果转换成依存句法树，从而能使用依存语法的语料库或者工具来改进链语法分析器
- 能更好地从其他语法理论（例如词语法（Word Grammar）[?]）角度来解析链语法分析结构
- 能简化一些 RelEx 中需要通过追溯很多链来找到中心词的规则

对于句子的中心词定义，语言学界有很多不同的看法。我们这里借鉴依存语法的观点，选择使用句子的主动词来作为句子的中心词，以及子句的主动词来作为子句的中心词。

此项改进的具体目标就是使句子的中心词，即句子的主动词，更容易从链语法分析得到的结果中被检索到。这样的改进不仅仅使链语法分析结构能够很直接地转换成类似依存句法树，因为依存语法一般以主动词为根节点；还无意中改进了链语法分析器对复句的分析能力，使其产生了更直观合理的分析结果，因为我们采用了将功能词与子句的主动词相连的方法来追溯子句的中心词，这显然比标准链语法中，不论什么情况，都使用功能词与子句的第一个词互相链接更直观合理；这样的改进给我们后续研究的依存关系抽取和逻辑关系抽取带来了诸多方便，因为在改进之前，无聊是并列复句还是偏正复句，或者是带状语的单句，链语法都将其中功能词与子句的第一个词相连，这样使得判断子句之间的语义逻辑关系变得很困难。

为了能实现这样的改进，我们首先引入了三种标准链语法中不存在的链接类型：WV，CV 和 IV。其中，WV 用于链接 LEFT-WALL 和句子的中心词，CV 用于链接功能词和子句的中心词，IV 用于链接 LEFT-WALL 和非限定动词。这些链接类型并不是第一批用来处理句子中心词的链接类型，链语法词典中有几个链接类型已经充当类似的角色，如 B，AF，CP,Eq，COq 等，但是这些链类型的用法非常复杂而且分类很模糊，引入的这几个新的链接类型，使句子的中心词角色更突出和直观，同时也和其他语法接轨。有关不同的链接语法中的链接类型的含义和使用方法，可参考链语法词典的在线文档：`http://www.abisource.org/projects/link-grammar/dict/section-WV.html`

对这一改进，我们采用的具体实现方法可以归纳为以下几步：

(a) 修改并增强链接 B，使得 B 能指向句子的中心词，以及子句的中心词。上面一段提到，链语法词典中的链接类型 B 有类似能指向句子的主动词的功能，但是其使用方法和能被应用的情况分类很模糊。

(b) 通过链接 B 找到句子的中心词，然后将 LEFT-WALL 和该中心词之间用新引入的 W 链接相连。

(c) 通过 B 链接找到子句的中心词，修改链接 CO 或者 Cs（CO 和 Cs 都是改进前链语法中用于链接功能词和子句的第一个词的链接类型），使其指向子句的中心词，并将其链接类型改为 CV。

IV 的处理方式和 WV 类似，这里不再详述。

下面给出了针对这些关系链接的改进的例子，例句：”Call me when you are ready.” 改进前句法分析结果如下：

```
    +------------------Xp------------------+
    |        +---MVs--+                    |
    +---Wi---+-Ox-+   +-Cs+-Spx+--Pa--+    |
    |        |    |   |   |    |      |    |
LEFT-WALL call.v me when you are.v ready.a .
```

改进后句法分析结果如下：

```
    +------------------Xp------------------+
    |        +---MVs--+---CV---+           |
    +---Wi---+-Ox-+   +   +-Spx+--Pa--+    |
    |        |    |   |   |    |      |    |
LEFT-WALL call.v me when you are.v ready.a .
```

例句：”I left soon after I saw you.” 改进前句法分析结果如下：

```
    +--------------------Xp--------------------+
    |             +----MVs----+                |
    +--Wd--+-Sp*i-+--MVa-+    +-Cs+Sp*i+-Ox-+  |
    |      |      |      |    |   |    |    |  |
LEFT-WALL I.p left.v-d soon when I.p saw.w you .
```

改进后句法分析结果如下：

```
    +--------------------Xp--------------------+
    +------WV-----+----MVs----+---CV---+       |
    +--Wd--+-Sp*i-+--MVa-+    +   +Sp*i+-Ox-+  |
    |      |      |      |    |   |    |    |  |
LEFT-WALL I.p left.v-d soon when I.p saw.w you .
```

例句：”Apparently, she loves cheese.” 改进前句法分析结果如下：

```
    +--------------------Xp------------------+
```

```
    +---------Wd--------+                    |
    |          +---CO---+                    |
    |          +--Xc-+  +--Ss-+----Ou---+     |
    |          |     |  |     |         |     |
LEFT-WALL apparently , she loves.v cheese.n-u .
```

改进后句法分析结果如下:

```
    +--------------------Xp------------------+
    +------------WV-----------+               |
    +---------Wd--------+     |               |
    |          +---CO---+     |               |
    |          +--Xc-+  +--Ss-+----Ou---+     |
    |          |     |  |     |         |     |
LEFT-WALL apparently , she loves.v cheese.n-u .
```

需要说明的是，在链语法分析器的当前版本中，虽然我们采用了 WV 链来链接 LEFT-WALL 和句子的中心词，但我们仍然保留了原来版本中的链接 LEFT-WALL 和句子的第一个词的 Wd 或者 Wi 等链，只是为了保持版本的向后兼容性。在转换成句法分析树的时候，我们会忽略 Wd 或者 Wi 等这些链。

5.1.3 RelEx

这一节我们将介绍依存关系抽取工具 RelEx（Relation Extractor）的工作原理和基本实现方法。RelEx 采用上一节中链语法分析器的输出作为输入，将其转换成一个特殊的特征结构图，然后根据相应的规则进行一系列的图结构转换，最后得到一个含有比链语法分析结果更抽象一点的介于句法关系和语义关系之间（syntactico-semantic）的依存关系图。

RelEx 包含了很多个模块，我们这里只介绍其中关键的模块。它的核心思想就是将链语法分析器的分析结果转换成一个特有的特征结构有向图（Feature Structure），然后使用一系列规则（在 RelEx 系统中被称为句子算法，Sentence Algorithm）对特征结构图进行一系列相应的有序的修改，最终得到一个精炼的特征结构图。最终得到的特征结构图中包含了句子中词和词之间的 RelEx

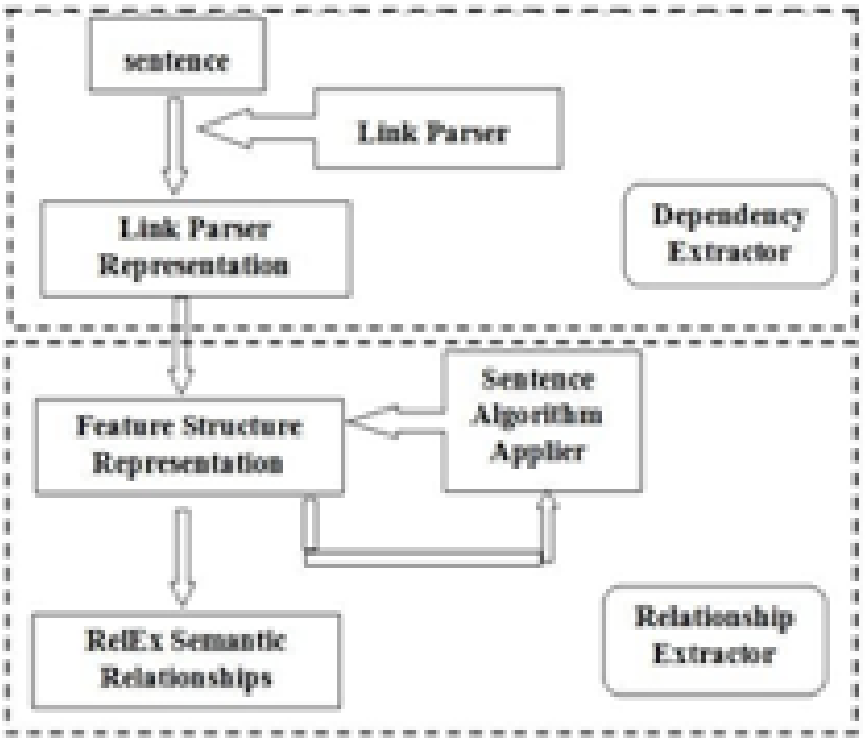


图 5.2　Overview of the internal dynamics of the RelEx semantic extraction system, which maps link parses into sets of more abstract syntactico-semantic relationships.

句法语义关系（如主谓关系、动宾关系等），还包含了句子中每个词的相关属性（如词性、时态等）。RelEx 还处理一些消歧工作，如指代消解，我们会在下面章节进一步阐述。

RelEx 的输出可看作是被简化了的输入句子的语法结构，在一定程度上，RelEx 还对一些表层关系进行归一化处理，因为很多等效但异构的动词框架会被映射到一致或者同态的特征结构图中。例如下面的两个不同的句子语义却一样：

```
Mary ate the cake.
The cake was eaten by Mary.
```

在 RelEx 的输出中，它们同时含有下面的 RelEx 依存关系：

```
_subj(ate, Mary)
_obj(ate, cake)
```

5.1.4 RelEx 的系统框架

RelEx 系统框架主要包括两大模块：附属关系抽取模块和语法关系抽取（参见图5.2）。它能识别句子中词和词之间的主语、宾语、间接宾语和其他依存关系，也能像其他依存语法分析器那样生成依存关系树。

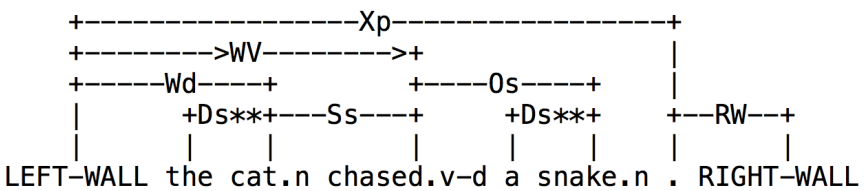
RelEx 首先对句子中的每个词创建一个特征节点（FeatureNode），然后通过一系列相关规则不断更新每个特征节点。这些规则将链语法分析器的输出结果中的不同链的组合转换成相应的 RelEx 依存关系，有些转换可能是通过某一条规则直接从链到 RelEx 依存关系的转换，也可能是间接地根据几条规则动态修改一个词的特征节点，并结合其他相关的特征节点中的特征，最终得到相应的 RelEx 关系。

接下来我们使用在链语法章节中使用的例句“The cat chased a snake.”来更详细地解析 RelEx 的每个步骤和实现方法。

步骤 1：将链语法分析器的输出转换成一个特征结构图。

如上所述，RelEx 首先将链语法分析器的输出结果转换成一个特征结构图。特征结构图是一个带权有向图，其中节点可以表示一个值，也可以是一个无序的特征列表。RelEx 中，特征指的就是一条指向另一个节点的带权边，而特征的值通常是一个字符串。

从上一节中我们可知，例句输入链语法分析器后产生下面的链语法结构：



根据我们上面的步骤描述，首先需要将上面的链语法分析结构转换成特定的特征结构图。句子中的每个词都会被转换成特征结构图的一个节点，其中该节点包含以下特征（[]中的内容表示特征的类型，如 NEXT[node]表示该特征是一个特征节点类型）：

- NEXT[node]：该特征指向表示该词的后一个词（如果存在）的特征节

点。

- PREV[node]：该特征指向表示该词的前一个词（如果存在）的特征节点。
- this[node]：该特征指向自己（该特征的存在只是为了方便某些句子算法的执行）
- wall[node]：该特征指向表示 LEFT-WALL 的特征节点。
- index_in_sentence [int]: 该词在句子的位置，LEFT-WALL 的位置为 0，以此类推。
- start_char [int]: 该词的第一个字符在原句子中的位置。从 0 开始计算。
- end_char [int]: 该词的最后一个字符的后一个字符在原句中的位置。end_char - start_char = 字长
- str [string]: 该词的内容。该内容应该和从句子中取（start_char, end_char）之间的子串得到的结果一致。
- POS [string]: 该词的词性。
- num_left_links [int]: 从该词出发指向左方向的链接个数。
- num_right_links [int]: 从该词出发指向右方向的链接个数。
- linkL0 [node]: 指向表示从该词出发指向左方向的链接的特征节点
- linkL(1,2,3,...) [node]: 指向表示从该词出发指向左方向的链接的特征节点（如果从该词出发指向左方向的链接个数超过 1）
- linkR0 [node]: 指向表示从该词出发指向右方向的链接的特征节点
- linkR(1,2,3,...) [node]: 指向表示从该词出发指向右方向的链接的特征节点（如果从该词出发指向左方向的链接个数超过 1）

类似地，链语法分析结构中的每个链接也被表示成一个特征节点，其中该节点包含以下特征：

- LAB [string]: 该链接的名称

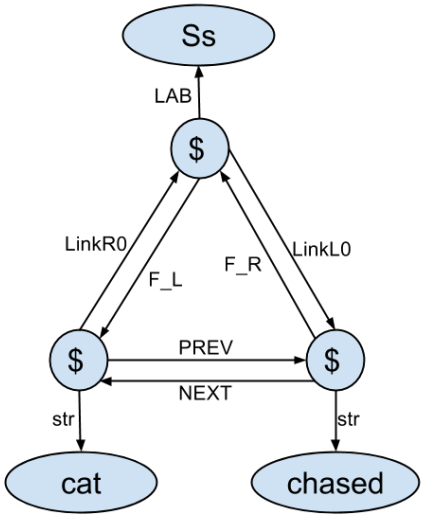


图 5.3 初始特征结构图

- F_L[node]: 指向表示该链接左边的词的节点
- F_R[node]: 指向表示该链接右边的词的节点

不难发现，RelEx 中的使用特征结构图非常复杂，为了能更形象地理解特征结构图，本文截取了上述例句的特征结构图的部分来进一步解释（参见图5.3）。

图5.3中，带$的节点可以表示代表句子中的某个词的特征节点，通过特征 str 指向词的内容；也可以表示代表某个链接的特征节点，通过特征 LAB 指向链接名称。从上图我们可以看出，“cat”和“snake”通过链接 Ss 相连。例句中的第二个个词节点的特征 str 表示该词的内容“cat”，特征 NEXT 指向表示下一个词的特征节点（即 str 为“chased”的节点），特征 F_L 表示该词是链接“Ss”的左右的词。“LinkR0”表示它指向的链接 Ss 是从该词“cat”出发指向右边的链接。以此类推，不难从上图中找出例句中第三个词节点的一系列相关特征。

步骤 2：执行包含一系列句子算法的句子算法应用器（Sentence Algorithm Applier），对步骤 1 中得到的特征结构图进行相应的改写操作。

句子算法应用器从句子算法的定义文件中导入一系列的句子算法，句子算法的执行顺序必须按照定义文件中句子算法的排序。当一个句子算法被执行时，RelEx 不断迭代遍历特征结构图中的每一个特征节点，对每个节点都尝试执行该句子算法。句子算法的执行会导致特征结构图的修改，因此它们的操作顺序是很重要的。例如，假如前面的句子算法执行后，特征结构中的所有特征节点的词性（POS）特征都被删除，那么用于处理含有 POS 特征节点的句子算法将无法被执行。

所有的句子算法都带有一个特定的标记（可以理解成该句子算法的名称），当某个句子算法被成功执行后，该算法会在被成功执行该算法的节点上添加一个特征 SIG，并将其赋值为该句子算法名称。被句子算法添加的一个最重要的特征是指向表示词的特征节点的 ref，RelEx 使用 ref 特征来输出最后的关系集合。

除了复杂的特征结构，RelEx 的另一个核心部分是前文一直提到的句子算法。RelEx 使用的句子算法也是很复杂的，可分为抽象的句子算法和需要实例化的句子算法。我们这里选择一个用的最广泛的通用句子算法 TemplateActionAlg 来进一步阐述。一个 TemplateActionAlg 实例能接受文本输入，第一行表示该实例的名称（不可重复），接下来是一个用于匹配的模板路径集合（Template Paths 和行为路径集合（Action Paths），之间用“=”隔开。对于模板路径<x, y, z>，表示目标值按照<x, y, z>这样的路径来解析。对于操作路径<x, y, z>，表示目标值按照<x, y, z>这样的路径被赋值。例如，在如下的被实例化的 TemplateActionAlg 的句子算法中，

```
#TemplateActionAlg
SPECIAL-ADJ
; Used for "easy to read."
; The B links back to the adjective. But it should really be
; interpretted as linking back to the subject of the adjective
<LAB> = \B\.*|\BW\.*
<F_L POS> = adj
=
<F_R obj> = %
```

```
<F_R obj> = <F_L subj>
<F_L ADJ-OBJ-FLAG> = T
```

在该句子算法中，模板就是：

```
<LAB> = \B\.*|\BW\.*
<F_L POS> = adj
```

不难看出，第一行表示用于匹配那些表示链接名称 B 或者 BW 的特征节点，句子算法的表示使用了正则表达式，意味着，只要链接名称以 B 或者 BW 的特征节点都满足。除此之外，要想上面的句子算法能被执行，还要满足第二行，即该特征节点表示的链接左边的词必须是一个形容词。

如果上面的两条模板都匹配成功，那么下面的操作将会被执行：

```
<F_R obj> = %
<F_R obj> = <F_L subj>
<F_L ADJ-OBJ-FLAG> = T
```

这些操作可被解释如下：第一行的操作表示该链接的右边的词所在的特征节点的 obj 特征会被清空。清空后，紧接着将该链接左边的词所在的特征节点的 sub 特征值赋给它。最后，将该链接的左边的词所在的特征节点的 ADJ-OBJ-FLAG 特征值设为 T（真）。

我们接着用上面的例句“The cat chased a snake.”，接下来两个图是图5.4的节选特征结构图，更直观地解释了，通过执行相关的句子算法后，被修改的结构特征图。

图5.4中虚线部分是执行了下面的句子算法后特征结构图被修改的部分：

```
#TemplateActionAlg
BASIC_REF
<str> = $string
=
<ref name> = <str>
```

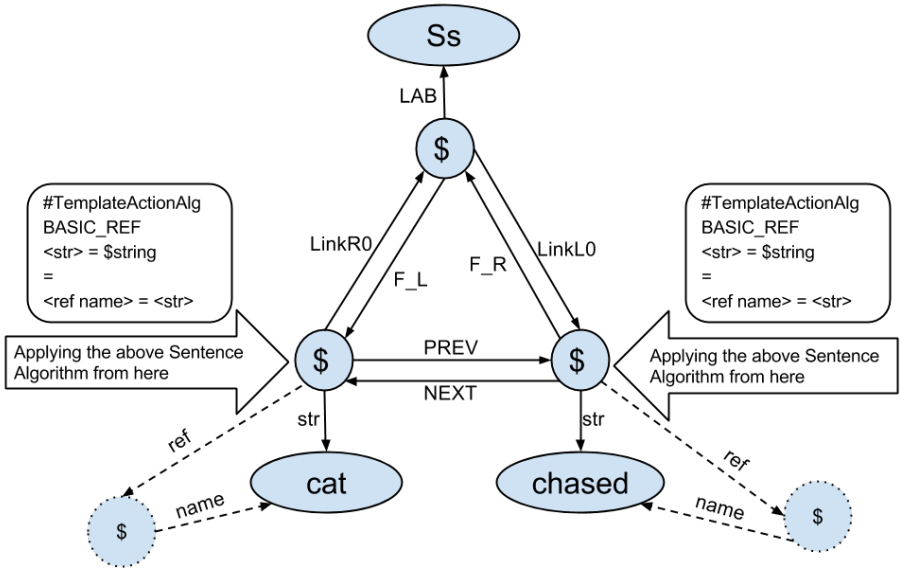


图 5.4　修改过的特征结构图

根据上面的句子算法，当模板“<str> = $string”被匹配时，会执行操作“<ref name>=<str>”，也就是，把该词的内容赋值给通过路径<ref, name>找到的节点。

图5.5中，同样地，虚线部分表示执行相应的句子算法后特征结构图被修改的部分。其中使用的句子算法有：

```
#TemplateActionAlg
STANDARD_SUBJ
<LAB> = \S\.*|\SX\.*|\Mg\.*|\MX\.*
<LAB> != \MX.[rj]\.*
<LAB> != MX|MXs|MXp
=
<IS_SUBJ_LINK> = T
```

如果找到连接主语的链接，那么创建一个名为 sub 的依存关系：

```
#TemplateActionAlg
MAKE_STANDARD_SUBJ
<IS_SUBJ_LINK> = T
```

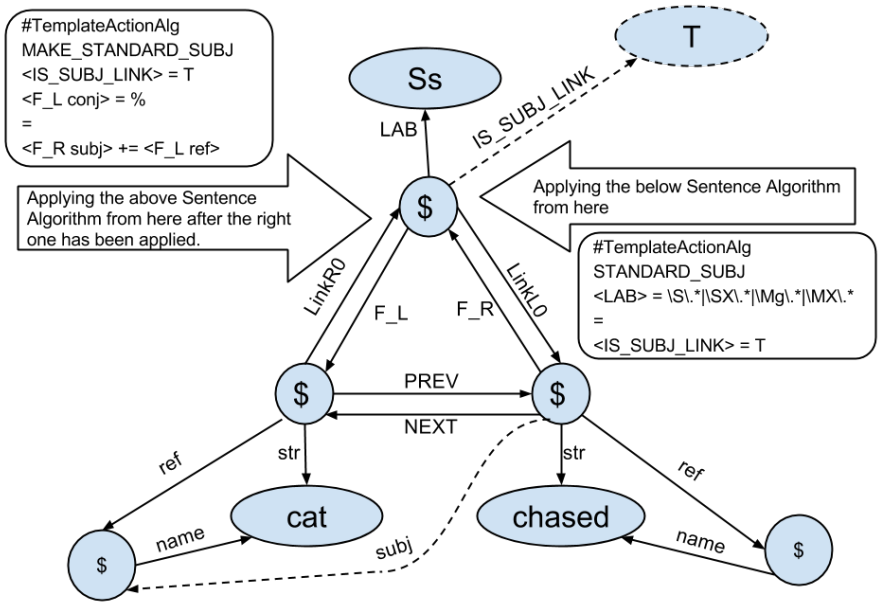


图 5.5 修改过的特征结构图

```
<F_L conj> = %
=
<F_R subj> += <F_L ref>
```

第一个句子算法中表明，当找到一个特征节点的“LAB”特征满足这样的正则表达式`\S\.*|\SX\.*|\Mg\.*|\MX\.*`时，即图5.5中最上面的特征节点的情况，那么会创建一个新的特征 IS_SUBJ_LINK 并赋值为”T”（真）。IS_SUBJ_LINK 是一个临时特征，会在上面第二个句子算法中用到，类似地，在执行了第一个句子算法后，图中最上面的节点也满足了第二个句子算法的条件，于是会接着执行第二个句子算法，即执行 $< F_R subj > + =< F_L ref >$，这说明，系统会将 $F_L ref$ 的值传到通过路径 $F_R subj$ 的相应特征上，也就是上图中最下面的虚线部分显示的。需要说明的是，操作符”+=” 是指将该操作符右边的值“附加”到通过左边的路径得到的特征上。

步骤 3：遍历最后得到的特征结构图，输出最终的 RelEx 关系集合。

当步骤 2 完成后，我们得到的结构特征图包含了很多 RelEx 句子算法中使用的临时特征，以及残留的句法特征，因此这一步骤主要是遍历步骤 2 得到的

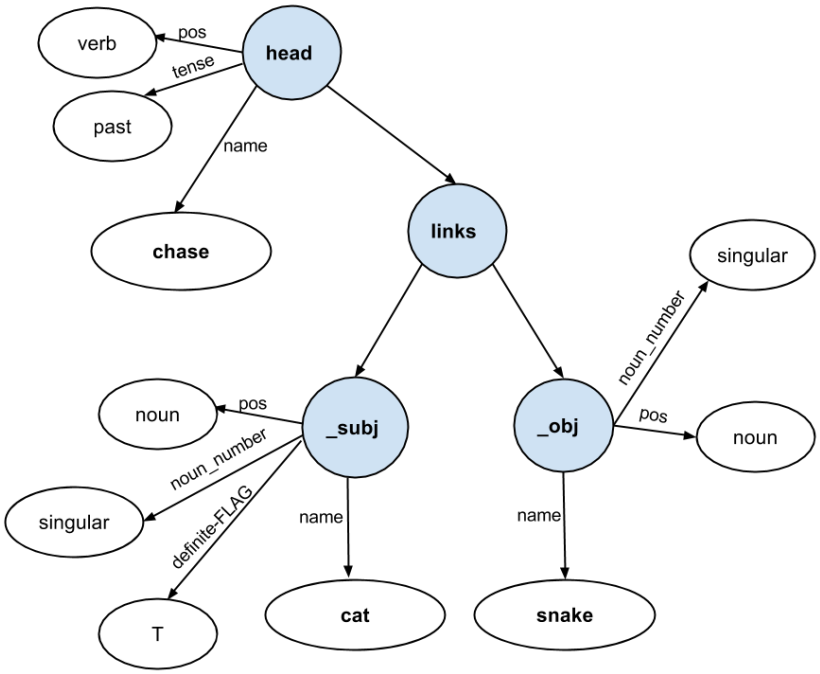


图 5.6　最终特征结构图

特征结构图并过滤掉这些不必要的特征，得到最终的简洁直观的 RelEx 关系树（图），最终抽取其中 RelEx 的依存关系和词语属性集合并输出。

这里依然采用上面的例句“The cat chased a snake.”。当所有的句子算法都被遍历过后，再过滤掉临时的特征，可以得到一个精简的用于生成 RelEx 关系集合的特征结构图（图5.6）。

其中特征节点 head 指向表示主句或所有从句中的中心词的特征节点。RelEx 中与 head 对应的还有特征节点 background（例句中没有），它是指向表示修辞词的节点，如形容词、副词等。

根据上图，我们可以从以下的特征节点抽取出每个词的相关属性：

- name: 指向表示该词的名称的特征节点
- pos: 指向表示该词的词性标记的特征节点
- tense: 指向表示该词的时态的特征节点（针对动词）
- definite-FLAG:指向一个值为“T”的特征节点，表明该名词受量词、定冠词等限定词约束，如“the””that”等（针对名词）

- noun_number: 指向表示该词的单复数形式的特征节点（针对名词）

类似，可以从以下的特征节点抽取出相关的依存关系：

- links: 指向表示以该节点为第一参数的依存关系的特征节点
- _subj: 指向表示依存关系_subj（主谓关系）的第二参数的特征节点
- _obj: 指向表示依存关系_obj（动宾关系）的第二参数的特征节点

根据图5.6中的信息，不难抽取最终的 RelEx 输出结果：

```
Dependency relations:

   _obj(chase, snake)
   _subj(chase, cat)

Attributes:

   tense(chase, past)
   subscript-TAG(chase, .v-d)
   pos(chase, verb)
   pos(., punctuation)
   subscript-TAG(snake, .n)
   pos(snake, noun)
   noun_number(snake, singular)
   definite-FLAG(cat, T)
   subscript-TAG(cat, .n)
   pos(cat, noun)
   noun_number(cat, singular)
   pos(a, det)
   pos(the, det)
```

其中，依存关系（Dependency relations）可以理解成两个词之间语义角色关系，如主谓关系、动宾关系等[1]。属性（Attributes）是值每个词的词态属性，如词性、时态、人称等[2]。

[1]RelEx 中使用的依存关系类型集合可参见http://wiki.opencog.org/w/Binary_relations
[2]RelEx 中使用的词态属性集合可参见http://wiki.opencog.org/w/Word_properties

5.1.5　RelEx 关系的形式化

如前所述，RelEx 关系反射了词和词之间的依存关系，如主谓关系、动宾关系、修饰和被修饰关系、所有格关系等等；除此之外，RelEx 也同样标注了词的各种词态属性，如时态、词性、单复数等等。为了这些繁琐的关系有一个适合操作以及后续处理的规范体系，本小节提出了一种基于巴克斯范式 BNF(Backus Normal Form)的对 RelEx 关系进行了如下的形式化表示：

```
<RelEx-rel>::== <dependency-rel> | <property-rel>
<dependency-rel>::== <dependency-type> "(" <word-instance> ", " <word-instance>")"
<property-rel> ::== <property-type> "(" <word-instance> ", "<property-value> ")"
<word-instance> ::== <word> <instance> | <word>
```

其中，

- RelEx-rel 可以是 RelEx 中的表示两个词之间的依存关系，也可以表示单个词的本身属性和属性值之间的关系。
- dependency-rel 表示 RelEx 中的依存关系，其中依次包含依存关系类型（即范式中使用的 dependency-type），第一参数（一般是单词，有时候也可以是短语）以及第二参数（一般是单词，有时候也可以是短语）
- property-rel 表示 RelEx 中对词的属性的标注，其中依次包含属性类型（即范式中使用的 property-type），词或短语以及属性值。
- word-instance 表示来自所分析的句子中的词，如果同一个词在同一个句子中出现 2 次或以上，通过在后面加上标记（用整数表示）来区分。

5.2　基于涉身的指代消解

RelEx 含有使用典型 Hobbs 算法实现了简单的指代消解[?]。概括地说，当识别出某语句中的”他”、”她”、”它”等代词，Hobbs 算法就会在近期使用过的语句中进行搜索，并根据数字、性别和其他特征来找出匹配此代词的名词。Hobbs 算法被用来建立一整列以近因排序的候选名词。

一个新的指代消解算法近期已在 OpenCogd 的 Atomspace 中实现，使用了类似 Hobbs 算法中指代消解的解决思路，但是使用了更灵活的知识表示，也能结合语义链接。本文实现了此算法，让一个以 OpenCog 控制的智能体生活在充斥着物体的环境，而每个物体都有它们自己的特性。此范例采用的是当前以 Atomspace 为基础、但较早期版本的指代消解系统，而有一段视频描述了这个范例`https://www.youtube.com/watch?v=ii-qdubNsx0`)。

举例来说，在一游戏世界中，我们可以有许多各种颜色和大小的球。我们运用多个节点将此呈现在 OpenCog Atomspace：表示"球"的概念的单个 ConceptNode、与"球"相关联的一个 WordNode、以及表示特定球的多个 ConceptNode。当然有些 ConceptNode 也表达其他与球有关的概念，像是"又鼓又大、会发出咯吱声的球"、"可用球棒挥击的球"等不概述于任何自然语言字词的概念。

当智能体与世界互动时，它会通过感知来学到寻获物体的相关信息。关联某设定物体的感知会贮存为其他表示特定物体实例的节点。所有此信息会使用 RelEx2Logic 被呈现在 Atomspace 之中（这部分在下一节会举例说明）。

举例来说，当用户说"抓红色的球"(" Grab the red ball")，智能体就必须理解用户指的是那种球 – 例如它必须唤起参照消解程序。参照消解使用句中的信息来选择物件、和些许捷思规则。概括而言，参照消解是根据智能体通过感知获取关于世界的知识，将用户句中的名词对应到虚拟世界的实际物体。

在这个例子中，首先智能体会选择相关"球"字眼的 ConceptNode，再利用句中的决定因素和其他适当的限制条件，检验所有与这些概念关联的个别实例（就此范例而已，决定因素为形容词"红色"；而由于动词为"抓"，因此它也会寻找可抓取的物体）。若它找到不止一个"可抓取的红球"，就会使用捷思规则来选取一个（在这个情况，智能体会选择离自己最近的物体）。

智能体也须将句中的代词对应到虚拟世界的实际物体。例如，若用户说"我喜欢红色的球。抓起它"(" I like the red ball. Grab it")，智能体就必须将代词"它"(" it") 对应到特定的红球。此程序分两个阶段执行：首先使用指代消解将代词"它"(" it") 与先前听到的名词"球"(" ball") 联系；接着使用参照消解将名词"球"(" ball") 与实际物体联系在一起。

在此情境中，我们可使用智能体对世界的知识改良 Hobbs 演算的结果，以帮助选出最佳的候选名词。假设智能体听到这些句子：

```
"球是红色的。"("The ball is red.")
"棍子是棕色的。"("The stick is brown.")
```

and then it receives a third sentence

```
"抓它。"("Grab it.")
```

指代消解器将构建一个列表，此列表包含针对第三个句子中代词“it”的两个选项：ball 和 stick。如果 stick 是最近使用的名词，那么正如 Hobbs 建议的那样，代理将抓取 stick，而不是 ball。

与此类似，如果智能体的历史包含

```
"From here I can see a tree and a ball."
"Grab it."
```

那么 Hobbs 的算法按顺序返回名词 tree 和 ball 作为候选。但是如果使用我们的集成的指代消解处理，智能体将推断不能抓取 tree，所以仅选择 ball。

5.3　RelEx2Logic：逻辑关系抽取

本章节介绍我们的自然语言理解系统中最后的纯语言学阶段，即 RelEx2Logic。RelEx2Logic 是我们本文工作的一个关键组成部分，用于将 RelEx 生成的句法-语义关系转换成 PLN 和其它 OpenCog 认知组件可以使用的基于超图的逻辑表示形式，从而使得我们的自然语言理解系统可以与逻辑推理系统（例如 PLN）、控制系统（例如 OpenPsi）和 OpenCog 的其它组件进行有效交互。

5.3.1　RelEx2Logic

众所周知，自然语言的逻辑极其复杂，而且即使在给定的表示形式下，同一个句子可以用很多种不同的逻辑表达方式，而不影响该句子想要表达的语义。

因此在我们的工作中，即通过 OpenCog 中的 Atomspace 来表示自然语言的语义关系和结构，必须面对这个事实：即使对于一个简单地句子，也可以有很多种不同结构的基于 Atomspace 的逻辑形式来相同或等价的语义。RelEx2Logic 系统的目标是以尽可能简单的方式将 RelEx 输出映射到 PLN 可以理解和推理的 AtomSpace 表示，生成可以获取相关表达所包含的大量含义的精确表示。RelEx2Logic 不会试图消解 RelEx 输出中所有的语义歧义，而是专注于将这些语义歧义表达成一种易于发现和处理的方式，这样能使这些歧义义在整个认知体系结构的知识库上或者借助涉身相关的信息，通过简单的常识推理而得到解决。

具体实现上，RelEx2Logic 的核心理念是通过应用一系列简单的重写规则以及较复杂的后处理规则（处理复杂句子和问题需要这些后处理规则，例如进行概念实例区分）把 RelEx 关系映射到语义解释（例如 OpenCog 的 AtomSpace 表示）。

每个核心重写规则的输入是句法分析图中满足特定约束的子图，输出是一个原子（Atom）超图。在实践中，需要的规则通常采用成对标记(G,A)的形式，其中：

- G 是一个图，其节点是单词或变量，其边是 RelEx 关系类型。
- A 是一个超图，其节点是单词、变量或特殊的语言学节点（根据一个小型词汇表绘制），其超图链接是 OpenCog 原子（Atom）类型（例如 InheritanceLink 和 EvaluationLink）。
- G 和 A 中的变量列表必须相同。

我们把符合这种说明的规则称为“简单映射规则”。

对于处理人类语言真正需要的简单映射规则来讲，满足约束条件**G 中的每个边映射到 A 中的单个超图链接**。从数学上讲，这个约束条件的后半部分意味着每个重写规则各自都是一个图同态[?]，这进一步意味着一起应用的一系列重写规则也是一个图同态。

这种规则的一个简单例子是这样一个(G,A)，其中

$$G = \{S_*(v_1, v_2),\ O_*(v_1, v_3)\}$$

$$A = (EvaluationLink\ v_1\ v_2\ v_3)$$

这把主语为 v_2，宾语为 v_3 的动词 v_1 映射到一个 v1 为谓语，(v_2, v_3)为参数列表的 OpenCog EvaluationLink。例如，S_* 指任何 S 的链路分析器子类型。当然，大部分规则都比这个例子复杂。

在具体操作中，由于自然语言的复杂性，仍然有一些这种形式的 RelEx2Logic 规则无法处理的语言现象。所以，为了获取句子正确的逻辑表示而不影响其含义，我们向 RelEx2Logic 框架添加一个后处理步骤，这个步骤主要包含下列部分：

- 处理 OpenCog Atomspace 中的实例。根据输入的自然语言的含义，将通用概念或谓语与特定概念或谓语进行区分。
- 估计模糊谓语或概念的概率。例如，如果句子是“Maybe the cat chased a snake.”，那么我们需要给最后的逻辑表示指定相较于句子“the cat chased a snake.”较低的概率。
- 正确处理量词。众所周知，自然语言中的量词十分复杂。不可能创建一个通用规则而以通用方式获取所有量词的正确抽象表示。而且，我们想要使得我们的规则更加通用，所以我们不想为每个量词使用单独的规则，这需要以简化的方式编写量词规则，而不是极力利用后处理方法允许的更大灵活性。

下面将解释后处理的更多细节。

将我们当前的简单映射规则应用于上述的例句“The cat chased a snake.”，会生成以下基本输出：

```
EvaluationLink
```

```
PredicateNode chase@3453432
ListLink
      ConceptNode cat@1243546464
      ConceptNode snake@564636322
```

这些“基本输出”仅仅是这个句子解释后 Atomspace 中所创建的完整原子（Atom）集合的一小部分。一个更完整的列表是

```
((ReferenceLink
   (InterpretationNode "sentence@f207bfb1-8dcf-42c8-a1d7-9d89075e8be8_parse_0_interpretation_$X
   (SetLink
      (ImplicationLink
         (PredicateNode "chased@64f680a3-d076-4ae1-814c-0152c3f3c8f8")
         (PredicateNode "chase" (ptv 1 0 1))
      )
      (InheritanceLink
         (ConceptNode "cat@3a186cb8-f8e2-40b1-97fc-525d51d29f57")
         (ConceptNode "cat" (ptv 1 0 1))
      )
      (InheritanceLink
         (ConceptNode "snake@b8aabaee-dfe9-4833-8b89-6d9c1d45aea0")
         (ConceptNode "snake" (ptv 1 0 1))
      )
      (EvaluationLink
         (PredicateNode "chased@64f680a3-d076-4ae1-814c-0152c3f3c8f8")
         (ListLink
            (ConceptNode "cat@3a186cb8-f8e2-40b1-97fc-525d51d29f57")
            (ConceptNode "snake@b8aabaee-dfe9-4833-8b89-6d9c1d45aea0")
         )
      )
      (InheritanceLink
         (PredicateNode "chased@64f680a3-d076-4ae1-814c-0152c3f3c8f8")
         (ConceptNode "past")
      )
      (InheritanceLink
         (InterpretationNode "sentence@f207bfb1-8dcf-42c8-a1d7-9d89075e8be8_parse_0_interpretat
         (ConceptNode "DeclarativeSpeechAct")
      )
      (EvaluationLink
```

```
            (PredicateNode "definite")
            (ListLink
               (ConceptNode "cat@3a186cb8-f8e2-40b1-97fc-525d51d29f57")
            )
         )
      )
   )
)
```

本节最后会给出更多映射例子以及所涉及原子（Atom）类型的解释。

如果链路分析器给出多个分析选项，RelEx2Logic 规则将为每个分析选项提供一个逻辑解释。目前已经实现基于语料库进行统计启发来排列多个分析的可信性，当然这并不完美。一些情况中，会向 OpenCog 提供多个逻辑输出选项，并且必须基于更高层次的上下文推断在这些选项之间进行选择，这是当前研究的一个难题和主题。

5.3.2　RelEx2Logic 所使用的规则

为了将 RelEx 关系转换成抽象的 OpenCog AtomSpace 逻辑表示，我们引入一系列 RelEx2Logic 规则。RelEx2Logic 规则（以前称为 Link2Atom 和 Syn2Sem 规则[?]）是一系列人工编码的规则，用于把 RelEx 的输出映射到 OpenCog 原子（Atom）表示的形式，这种表示更适合于由 PLN 和其它工具进行自动化推断。本节总结性地阐述规则集合并给出一些规则例子。可以在`opencog/nlp/scm/relex-to-logic.scm`在线找到所有规则。

RelEx2Logic 规则有多个目标：

- 作为运行的 OpenCog 自然语言处理系统的一部分，其潜在用途包括文本分析和自然语言对话。
- 作为概念研究工具，帮助我们理解句法-语义输出（例如 RelEx 生成的句法-语义输出）映射到更抽象的逻辑关系（例如 PLN 采用的逻辑关系）的问题。

- 作为工具来帮助生成成对（链路分析和抽象的 PLN 形式的原子）的语料库，然后可以自动分析此语料库来产生语言理解和生成规则（更多信息请参见 Syn2Sem）。

示例规则（Tense 规则）

为了方便阐述剩下的内容，我们从一个非常简单的 RelEx2Logic 规则例子开始。当然，大部分规则比这个例子复杂。此处，为了方便人理解，采用 RelEx SFF 格式（RelEx 许多输出格式中的一种）来说明规则输入。

为了用便于阅读的方式说明 RelEx2Logic 规则，我们在此使用类似以下的格式：

```
规则输入:
tense(W, Tense)
pos(W, verb)
规则输出:
(tense-rule W (get_instance_name W word_index sentence_index) Tense)
```

在 OpenCog 中运行此规则，规则的输出将启动一个帮助函数（helper function，用 Scheme 语言编写）。下面会列出这个帮助函数。然后帮助函数将最终生成最后的 OpenCog Atomspace 表示。

如同上一小节中所述，在我们软件实现中，我们使用诸如(ImplicationLink P_G P_A)这样的格式来表示 RelEx2Logic 规则，从而使得 RelEx2Logic 与 OpenCog 系统的其余组件保持一致。例如，这种格式意味着规则可以由 OpenCog 模式匹配器来执行（OpenCog 模式匹配器用来在 OpenCog Atomspace 中搜索原子的特定模式、排列或“模板”）。所以上述简单的 RelEx2Logic 规则在当前系统中将被如下表示：

```
ImplicationLink
     EvaluationLink
          PredicateNode "tense"
          ListLink
               WordNode W
```

```
            ConceptNode Tense
    ExecutionOutputLink
            GroundedSchemNode "r2l/tense-rule.scm"
            ListLink
                WordNode W
                ConceptNode Tense
                NumberNode word_index
                NumberNode sentence_index
```

对应于上述 RelEx2Logic 规则的提供后处理的相应帮助函数为：

帮助函数：

```
(define (tense-rule verb instance tense)
   (define new_predicate (PredicateNode instance))
   (define verb_node (PredicateNode verb))
   (define tense_node (ConceptNode tense))
   (list
      (InheritanceLink new_predicate verb_node)
      (InheritanceLink new_predicate tense_node)
   )
)
```

例子：

```
tense(chase, Past)
pos(chase, verb)
==>
(tense-rule "chase" "chase@3453432" "Past")
==>
(InheritanceLink "chase@3453432" chase)
(InheritanceLink "chase@3453432" Past)
```

注意：这个例子使用了额外的帮助函数 *get_instance_name*。

下面解释这个帮助函数。本质上，这个函数只是为概念的实例选择一个在给出的 AtomSpace 中唯一的名字（例如，它可能为 *chair* 的实例选择名字

*chair*77）。如果单词 W 在句子中是第 K 次出现，那么单词序号为 K。句子序号是该句子区别于其它句子的一个标识符。

5.3.3 RelEx2Logic 的后处理过程

RelEx2Logic 最初的设计意图是如同上述例子那样通过直接的逻辑含义进行所有需要的处理。但是随着工作的进展，我们认识到这样做在许多情况中会非常麻烦，同时我们发现引入一些更精巧的后处理方法是最直接的解决办法。此处我们阐述这些后处理规则的基本类型。

现阶段，我们已经实现的后处理规则包括下列类别：

- 量词：为带有量词的单词引入变量并通过使用相应的链路限定变量的范围，同时给链路指定一些真值。
- 模糊情态动词：为模糊情态动词发生过修改的谓语指定一些置信下限。例如“Maybe dogs like fish.”，这表示“dogs like fish”的置信度不是很高，所以我们把“he is a genius.”的置信度从默认的 0.99 调整为 0.5。
- 冗余实例清理：为可以从句法分析知道的通用概念清理实例号。例如“Dogs like meat.”，没有为 dogs 和 meat 指定任何明确的冠词、修饰语或代词，并且也没有意指任何其它概念，所以我们认为 dogs 和 meat 指通用概念 dog 和 meat，而不是某些特定概念（特定概念需要指定实例号）。

下面更详细地阐述我们怎样为上面列出的每种类别实现后处理。

量词：在前面的阶段，我们使用量词记号来标记需要进行后处理的量词。例如“All dogs like meat.”，在应用量词 RelEx2Logic 规则期间创建“allmarker”并生成下列结果。

```
EvaluationLink
      PredicateNode "allmarker"
       ListLink
             ConceptNode "dog@13456"
```

本例中，在后处理阶段，我们为量词“all”引入“ForAllLink”（它代表OpenCog Atomspace 中的通用量化）和变量节点“$X”来限定范围包含所有含有 *dog*@13456 的链路。然后，每个 *dog*@13456 都将被替换为 VariableNode“$X”。“allmarker”将在后处理后被删除。所以，在后处理结束后，将创建：

```
ForAllLink
      VariableNode "$X"
      ImplicationLink
            InheritanceLink "$X" noun_instance
            AndLink
                ** links involving "dog@13456" **
```

模糊情态动词：模糊情态动词后处理类似于量词后处理，我们在应用相关的RelEx2Logic 规则期间使用相应的记号来标记模糊情态动词。例如，对于句子“Maybe dogs like fish.”，在应用 Maybe RelEx2Logic 规则后生成下列表示：

```
EvaluationLink
       PredicateNode "maybemarker"
       ListLink
            PredicateNode "like@9768"
```

然后，用于后处理的 *maybemarker* 帮助函数将查找所有包含“like@9768”的根链路，并将置信度从默认的 0.99 调整为 0.5。

冗余实例清理：因为不想在句子转换成抽象语义表示期间丢失任何来自句法层面的信息，我们在后处理阶段设立“实例清理”步骤。在当前方法中，我们为句子中每个非组合单词创建一个单独的实例节点。但是，在很多情况中，这会生成许多不必要的实例节点。例如，在句子“Dogs like yummy meat.”中，无需创建诸如 *yummy*@1765 和 *dog*@2593 的实例节点，因为这个句子没有指定特定的“dog”和特殊的“yummy”。所以只创建类似以下的链路就已足够：

```
InheritanceLink
        ConceptNode   "meat@3976"
        ConceptNode   "yummy"
```

```
EvaluationLink
        PredicateNode "like"
        ListLink
            ConceptNode "dog"
            ConceptNode "meat@3976"
```

但是，一个诸如“meat@3976”的实例节点更加有用。因为在这个句子中，它仅陈述“Dogs like ’ YUMMY’ meat”，而并非一般的“meat”。

要想知道哪些单词真正需要实例节点，必须在一定程度上理解句子的结构。当前，我们使用一些常识性的规则开始这个过程。确切地讲，如果一个单词没有被修改（名词被形容词修改或者动词被副词修改），或者没有被定冠词限定，或者使用代词，那么此单词的实例节点为“冗余”。清理实例的后处理进行以下工作：

- 在 Atomspace 中（或者仅在 RelEx2Logic 规则的本地输出中）检查是否存在任何此名字的原子为“冗余”。
- 如果找到冗余原子，那么将其删除并将它的链路移至其父概念。

5.4 实验结果举例

本节中给出更多例子来解释我们的语言理解的理解管道，包含前面阐述的所有部分：Link Parser、RelEx 和 RelEx2Logic。在后面的章节中，这些特定例子也会在下一章中用来阐述基于自然语言的推理。

例 1：Socrates is a man.

链语法分析器输出：

```
    +--------------Xp-------------+
    +-------->WV------->+--Osm-+   |
    +----Wd----+---Ss---+  +Ds*+   +--RW--+
    |          |        |  |   |   |       |
LEFT-WALL Socrates[!] is.v a man.n . RIGHT-WALL
```

RelEx 输出：

```
Dependcy Relations:

    _obj(be, man)
    _subj(be, Socrates)

Attributes:

    tense(be, present)
    subscript-TAG(be, .v)
    pos(be, verb)
    pos(., punctuation)
    subscript-TAG(man, .n)
    pos(man, noun)
    noun_number(man, singular)
    definite-FLAG(Socrates, T)
    pos(Socrates, noun)
    noun_number(Socrates, singular)
    pos(a, det)
```

RelEx2Logic 输出：

```
((ReferenceLink
   (InterpretationNode "sentence@f04de205-0f67-471d-84fc-2f9c6103ebca_parse_0_interpretat
   (SetLink
      (InheritanceLink
         (ConceptNode "Socrates@cfbb87d6-5444-4845-a673-8d70b93ffde6")
         (ConceptNode "Socrates" (ptv 1 0 1))
      )
      (InheritanceLink
         (ConceptNode "man@fe411437-7607-4169-abcf-d42ebcd50338")
         (ConceptNode "man" (ptv 1 0 1))
      )
      (InheritanceLink
         (ConceptNode "Socrates@cfbb87d6-5444-4845-a673-8d70b93ffde6")
         (ConceptNode "man@fe411437-7607-4169-abcf-d42ebcd50338")
      )
```

```
      (ImplicationLink
         (PredicateNode "is@b06a386e-0983-4bc8-bd1e-395b487faff2")
         (PredicateNode "be" (ptv 1 0 1))
      )
      (InheritanceLink
         (PredicateNode "is@b06a386e-0983-4bc8-bd1e-395b487faff2")
         (ConceptNode "present")
      )
      (InheritanceLink
         (InterpretationNode "sentence@f04de205-0f67-471d-84fc-2f9c6103ebca_parse_0_interpretat
         (ConceptNode "DeclarativeSpeechAct")
      )
      (EvaluationLink
         (PredicateNode "definite")
         (ListLink
            (ConceptNode "Socrates@cfbb87d6-5444-4845-a673-8d70b93ffde6")
         )
      )
   )
)
)
```

例 **2:** Men breathe air.

链语法分析器输出：

```
     +---------------Xp-------------+
     +------->WV------>+             |
     +---Wd--+---Sp--+---Ou---+     +--RW--+
     |       |        |        |     |      |
LEFT-WALL men.p breathe.v air.n-u . RIGHT-WALL
```

RelEx 输出：

```
Dependcy Relations:

    _obj(breathe, air)
    _subj(breathe, man)
```

```
Attributes:

    tense(breathe, present)
    subscript-TAG(breathe, .v)
    pos(breathe, verb)
    pos(., punctuation)
    subscript-TAG(air, .n-u)
    pos(air, noun)
    noun_number(air, uncountable)
    subscript-TAG(man, .p)
    pos(man, noun)
    noun_number(man, plural)
```

RelEx2Logic 输出：

```
((ReferenceLink
   (InterpretationNode "sentence@05b9ba83-e424-43c8-b5f1-8261217bc548_parse_0_interpretat
   (SetLink
      (ImplicationLink
         (PredicateNode "breathe@ba9dec0b-b99f-4c2c-a5cc-31cc9aa4f172")
         (PredicateNode "breathe" (ptv 1 0 1))
      )
      (InheritanceLink
         (ConceptNode "men@5f6e723d-e57e-47e5-89df-ee9daaa2936d")
         (ConceptNode "man" (ptv 1 0 2))
      )
      (InheritanceLink
         (ConceptNode "air@98afd45c-4807-4607-a732-298c7979327f")
         (ConceptNode "air" (ptv 1 0 1))
      )
      (EvaluationLink
         (PredicateNode "breathe@ba9dec0b-b99f-4c2c-a5cc-31cc9aa4f172")
         (ListLink
            (ConceptNode "men@5f6e723d-e57e-47e5-89df-ee9daaa2936d")
            (ConceptNode "air@98afd45c-4807-4607-a732-298c7979327f")
         )
      )
      (InheritanceLink
```

```
        (PredicateNode "breathe@ba9dec0b-b99f-4c2c-a5cc-31cc9aa4f172")
        (ConceptNode "present")
      )
      (InheritanceLink
        (InterpretationNode "sentence@05b9ba83-e424-43c8-b5f1-8261217bc548_parse_0_interpretat
        (ConceptNode "DeclarativeSpeechAct")
      )
   )
)
)
```

例 3: Socrates breathes air.

链语法分析器输出：

```
    +-----------------Xp-----------------+
    +---------->WV---------->+           |
    +----Wd----+-----Ss----+---Ou---+    +--RW--+
    |          |           |        |    |      |
LEFT-WALL Socrates[!] breathes.v air.n-u . RIGHT-WALL
```

RelEx 输出：

```
Dependcy Relations:

    _obj(breathe, air)
    _subj(breathe, Socrates)

Attributes:

    tense(breathe, present)
    subscript-TAG(breathe, .v)
    pos(breathe, verb)
    pos(., punctuation)
    subscript-TAG(air, .n-u)
    pos(air, noun)
    noun_number(air, uncountable)
    definite-FLAG(Socrates, T)
    pos(Socrates, noun)
    noun_number(Socrates, singular)
```

RelEx2Logic 输出：

```
((ReferenceLink
   (InterpretationNode "sentence@a41e5f83-4a87-4c3f-ad39-833510272d90_parse_0_interpretat
   (SetLink
      (ImplicationLink
         (PredicateNode "breathes@0a04573a-c867-458d-baa9-0f347ee73371")
         (PredicateNode "breathe" (ptv 1 0 2))
      )
      (InheritanceLink
         (ConceptNode "Socrates@a34bbdf7-1dac-4e62-9dfa-48e1a1f9710c")
         (ConceptNode "Socrates" (ptv 1 0 2))
      )
      (InheritanceLink
         (ConceptNode "air@016cb5c7-35b7-42c4-8690-e3958030773f")
         (ConceptNode "air" (ptv 1 0 2))
      )
      (EvaluationLink
         (PredicateNode "breathes@0a04573a-c867-458d-baa9-0f347ee73371")
         (ListLink
            (ConceptNode "Socrates@a34bbdf7-1dac-4e62-9dfa-48e1a1f9710c")
            (ConceptNode "air@016cb5c7-35b7-42c4-8690-e3958030773f")
         )
      )
      (InheritanceLink
         (PredicateNode "breathes@0a04573a-c867-458d-baa9-0f347ee73371")
         (ConceptNode "present")
      )
      (InheritanceLink
         (InterpretationNode "sentence@a41e5f83-4a87-4c3f-ad39-833510272d90_parse_0_inter
         (ConceptNode "DeclarativeSpeechAct")
      )
      (EvaluationLink
         (PredicateNode "definite")
         (ListLink
            (ConceptNode "Socrates@a34bbdf7-1dac-4e62-9dfa-48e1a1f9710c")
         )
```

```
      )
   )
)
)
```

例 4: I think Socrates is a man.

链语法分析器输出：

```
    +--------------------Xp--------------------+
    +---->WV---->+--------CV------>+--Osm-+    |
    +--Wd--+-Sp*i+----Ce---+---Ss---+  +Ds*+   +--RW--+
    |      |     |         |        |  |   |   |      |
LEFT-WALL I.p think.v Socrates[!] is.v a man.n . RIGHT-WALL
```

RelEx 输出：

```
Dependcy Relations:

    _obj(be, man)
    _subj(be, Socrates)
    _rep(think, be)
    _subj(think, I)

Attributes:

    definite-FLAG(Socrates, T)
    pos(Socrates, noun)
    noun_number(Socrates, singular)
    pos(., punctuation)
    subscript-TAG(man, .n)
    pos(man, noun)
    noun_number(man, singular)
    pos(a, det)
    tense(be, present)
    HYP(be, T)
    subscript-TAG(be, .v)
    pos(be, verb)
    tense(think, present)
    subscript-TAG(think, .v)
```

```
pos(think, verb)
pronoun-FLAG(I, T)
gender(I, person)
definite-FLAG(I, T)
subscript-TAG(I, .p)
pos(I, noun)
noun_number(I, singular)
```

RelEx2Logic 输出：

```
((ReferenceLink
   (InterpretationNode "sentence@ec1ca4d5-20ce-465c-9686-aa3edd4dbb2f_parse_0_interpretat
   (SetLink
      (ImplicationLink
         (PredicateNode "think@8be218e8-095b-44e7-b321-ee877fe4ad45")
         (PredicateNode "think" (ptv 1 0 1))
      )
      (InheritanceLink
         (ConceptNode "I@8c841336-dc92-4c3a-a5af-2e860365ff88")
         (ConceptNode "I" (ptv 1 0 1))
      )
      (EvaluationLink
         (PredicateNode "think@8be218e8-095b-44e7-b321-ee877fe4ad45")
         (ListLink
            (ConceptNode "I@8c841336-dc92-4c3a-a5af-2e860365ff88")
         )
      )
      (ImplicationLink
         (PredicateNode "is@14df1d2f-f71e-45c9-b4a3-80eb38c6a89f")
         (PredicateNode "be" (ptv 1 0 2))
      )
      (EvaluationLink
         (PredicateNode "think@8be218e8-095b-44e7-b321-ee877fe4ad45")
         (ListLink
            (ConceptNode "is@14df1d2f-f71e-45c9-b4a3-80eb38c6a89f")
         )
      )
```

```
        (InheritanceLink
           (PredicateNode "think@8be218e8-095b-44e7-b321-ee877fe4ad45")
           (ConceptNode "present")
        )
        (InheritanceLink
           (ConceptNode "Socrates@c1fde686-ae83-45e8-a60d-259179597e68")
           (ConceptNode "Socrates" (ptv 1 0 3))
        )
        (EvaluationLink
           (PredicateNode "definite")
           (ListLink
              (ConceptNode "Socrates@c1fde686-ae83-45e8-a60d-259179597e68")
           )
        )
        (InheritanceLink
           (InterpretationNode "sentence@ec1ca4d5-20ce-465c-9686-aa3edd4dbb2f_parse_0_interpretat
           (ConceptNode "DeclarativeSpeechAct")
        )
        (InheritanceLink
           (ConceptNode "man@aed74351-8482-4a5b-a6f4-3e979b537fae")
           (ConceptNode "man" (ptv 1 0 3))
        )
        (InheritanceLink
           (ConceptNode "Socrates@c1fde686-ae83-45e8-a60d-259179597e68")
           (ConceptNode "man@aed74351-8482-4a5b-a6f4-3e979b537fae")
        )
        (InheritanceLink
           (PredicateNode "is@14df1d2f-f71e-45c9-b4a3-80eb38c6a89f")
           (ConceptNode "present")
        )
        (EvaluationLink
           (PredicateNode "definite")
           (ListLink
              (ConceptNode "I@8c841336-dc92-4c3a-a5af-2e860365ff88")
           )
        )
     )
  )
```

)

例 5: Bob thinks Socrates is a woman.

链语法分析器输出:

```
    +----------------------Xp----------------------+
    +------>WV----->+--------CV------->+--Ost--+    |
    +---Wd--+---Ss--+----Ce---+---Ss---+  +Ds**+    +--RW--+
    |       |       |         |        |  |    |    |      |
LEFT-WALL Bob.m thinks.v Socrates[!] is.v a woman.n . RIGHT-WALL
```

RelEx 输出:

```
dependcy Relations:

    _obj(be, woman)
    _subj(be, Socrates)
    _rep(think, be)
    _subj(think, Bob)

Attributes:

    definite-FLAG(Socrates, T)
    pos(Socrates, noun)
    noun_number(Socrates, singular)
    pos(., punctuation)
    subscript-TAG(woman, .n)
    pos(woman, noun)
    noun_number(woman, singular)
    pos(a, det)
    tense(be, present)
    HYP(be, T)
    subscript-TAG(be, .v)
    pos(be, verb)
    tense(think, present)
    subscript-TAG(think, .v)
    pos(think, verb)
    gender(Bob, masculine)
    definite-FLAG(Bob, T)
    person-FLAG(Bob, T)
```

```
subscript-TAG(Bob, .m)
pos(Bob, noun)
noun_number(Bob, singular)
```

RelEx2Logic 输出：

```
((ReferenceLink
   (InterpretationNode "sentence@8c6d9ca7-abc8-4dbd-8442-669e62a32cc0_parse_0_interpretation_$X
   (SetLink
      (ImplicationLink
         (PredicateNode "thinks@87f3ac28-56e7-44fc-8125-815b2e6747ee")
         (PredicateNode "think" (ptv 1 0 2))
      )
      (InheritanceLink
         (ConceptNode "Bob@f16c5987-7193-4809-9a3f-94c73a77d4d4")
         (ConceptNode "Bob" (ptv 1 0 1))
      )
      (EvaluationLink
         (PredicateNode "thinks@87f3ac28-56e7-44fc-8125-815b2e6747ee")
         (ListLink
            (ConceptNode "Bob@f16c5987-7193-4809-9a3f-94c73a77d4d4")
         )
      )
      (ImplicationLink
         (PredicateNode "is@712b5da0-e4df-4b24-9bca-e0a1114a6854")
         (PredicateNode "be" (ptv 1 0 3))
      )
      (EvaluationLink
         (PredicateNode "thinks@87f3ac28-56e7-44fc-8125-815b2e6747ee")
         (ListLink
            (ConceptNode "is@712b5da0-e4df-4b24-9bca-e0a1114a6854")
         )
      )
      (InheritanceLink
         (PredicateNode "thinks@87f3ac28-56e7-44fc-8125-815b2e6747ee")
         (ConceptNode "present")
      )
```

```
(InheritanceLink
   (ConceptNode "Socrates@eadf1215-ca8d-42ee-88b9-ac1e6471f8c1")
   (ConceptNode "Socrates" (ptv 1 0 4))
)
(EvaluationLink
   (PredicateNode "definite")
   (ListLink
      (ConceptNode "Socrates@eadf1215-ca8d-42ee-88b9-ac1e6471f8c1")
   )
)
(InheritanceLink
   (InterpretationNode "sentence@8c6d9ca7-abc8-4dbd-8442-669e62a32cc0_parse_0_inter
   (ConceptNode "DeclarativeSpeechAct")
)
(InheritanceLink
   (ConceptNode "woman@9d719ddc-0fe5-4171-a4e5-5397a6c6e9e9")
   (ConceptNode "woman" (ptv 1 0 1))
)
(InheritanceLink
   (ConceptNode "Socrates@eadf1215-ca8d-42ee-88b9-ac1e6471f8c1")
   (ConceptNode "woman@9d719ddc-0fe5-4171-a4e5-5397a6c6e9e9")
)
(InheritanceLink
   (PredicateNode "is@712b5da0-e4df-4b24-9bca-e0a1114a6854")
   (ConceptNode "present")
)
(InheritanceLink
   (SpecificEntityNode "Bob@f16c5987-7193-4809-9a3f-94c73a77d4d4")
   (ConceptNode "male")
)
(InheritanceLink
   (SpecificEntityNode "Bob@f16c5987-7193-4809-9a3f-94c73a77d4d4")
   (ConceptNode "Bob" (ptv 1 0 1))
)
(EvaluationLink
   (PredicateNode "definite")
   (ListLink
      (ConceptNode "Bob@f16c5987-7193-4809-9a3f-94c73a77d4d4")
```

```
        )
      )
    )
)
)
```

5.5 比较级的处理：实例分析

比较级提供了从表层形式映射到我们设计和实现的逻辑表示的有趣例子。

关于正确处理英语和其它语言中的比较级，理论上的语言学从来没有达成一致。一些理论家假定一种省略理论，建议从句子的表层结构得出比较级语法，忽略深层结构中存在的某些单词[?][?]。另外一些理论家假定一种移动理论[?][?]，这种理论更多地由传统的生成语法启示，假设比较级语法包含一个重新排列深层结构的表层结构。

链路语法框架从根本上回避了这种问题：不管省略理论还是移动理论，都由链路语法词典中的某些对称性表示，但是这些对称性不需要被链路分析器本身显式地识别或使用（虽然这些对称性可以指导人类或 AI 系统创建链路语法词典）。从目前的经验上讲，链路分析器对比较级的处理相当好，但是相关的词典条目有点混乱并且实际上并不完全对称。这说明两种可能：

(a) 英语比较级的语法“复杂”而混乱，不适用任何可用的理论。并且/或者

(b) 链路语法词典可以在比较级方面进行极大的改进

我们猜测事实是两者兼而有之。但是请注意，将链路语法作为用于理解复杂句子（包含比较级）的实际管道的一部分而部署时，我们不需要决定这个问题。

我们通过一个例子说明我们的框架如何处理比较级。RelEx2Logic 的一个用于比较级的规则，其简短形式如下：

```
than(w1, w2)
_comparative(ad, w)
==>
```

```
TruthValueGreaterThanLink
    InheritanceLink w1 ad
    InheritanceLink w2 ad
```

图5.7给出了完整形式。

使用这个规则的一个简单例子是：

```
Pumpkin is cuter than the white dog.
==>
_predadj(cute, Pumpkin)
than(Pumpkin, dog)
_comparative(cute, Pumpkin)
_amod(dog, white)
==>
AndLink
   InheritanceLink dog_11 white
   InheritanceLink dog_11 dog
   TruthValueGreaterThanLink
       InheritanceLink Pumpkin cute
       InheritanceLink dog_11 cute
```

另一方面，为了处理诸如“Amen is more intelligent than insane”这样的句子，我们使用一个不同的规则，这个规则的简短形式如下：

```
_predadj(adj1, W)
than(adj1, adj2)
_comparative(adj1, W)
==>
TruthValueGreaterThanLink
    InheritanceLink W adj1
    InheritanceLink W adj2
```

结果输出如下：

```
_predadj(intelligent, Amen)
than(intelligent, insane)
_comparative(intelligent, Amen)
==>
TruthValueGreaterThanLink
```

INPUT:

_comparative(adj/adv, w1)

than(w1, w2)

OUTPUT:

(comparative-rule W1 (get_instance_name W1 word_index sentence_index) W2 (get_instance_name W2 word_index sentence_index) AD (get_instance_name AD word_index sentence_index))

HELPER FUNCTION:

```
(define (comparative-rule w1 w1_instance w2 w2_instance ad ad_instance)
  (define new_concept_1 (ConceptNode w1_instance))
   (define new_concept_2 (ConceptNode w2_instance))
   (define word_node_1 (ConceptNode w1))
   (define word_node_2 (ConceptNode w2))
   (define ad_node (ConceptNode ad))
   (define new_ad_node (ConceptNode ad_instance))
   (AndLink
       (InheritanceLink new_ad_node ad_node)
       (InheritanceLink new_concept_1 word_node_1)
       (InheritanceLink new_concept_2 word_node_2)
       (TruthValueGreaterThanLink
             (InheritanceLink new_concept_1 new_ad_node)
             (InheritanceLink new_concept_2 new_ad_node)
       )
   )
)
```

图 5.7 Full RelEx2Logic rule for parsing one type of comparative sentence.

```
InheritanceLink Amen intelligent
InheritanceLink Amen insane
```

5.6　在简单句子集的性能评价

由于本文中的自然语言理解系统最终输出的基于超图逻辑形式比较独特，无法与其他类似系统进行比较。这里我们从路透社的语料库中随机抽取了 100 个词数小于 8 的句子作为测试集（选择相对较短句子来作为测试集，我们的出发点在于着重测试该自然语言理解系统对词语或短语之间的语义关系，而非专注于处理复杂的语言现象），在上面运行了我们的自然语言处理系统，并对输出结果做了人工分析，评价结果可参见下表：

	Link Parser
完全正确	96
几乎正确	2
错误	2

表 5.1　改进后的链语法分析器在该测试集上的分析结果。“几乎正确”是指分析结果中忽略了链语法词典中不存在的词，但对剩下部分解析正确。

	RelEx
完全正确	80
几乎正确	15
错误	1

表 5.2 在链语法分析器解析完全正确的 96 个句子上，RelEx 的性能分析。其中“几乎正确”是指输出中忽略或错误输出了 1 个以内的依存关系。

	RelEx2Logic
完全正确	68
几乎正确	12
错误	0

表 5.3 在链语法分析器和 RelEx 都解析完全正确的 80 个句子上，RelEx2Logic 的性能分析。其中“几乎正确”是指输出中忽略或错误输出了 1 个以内的语义关系。’

在该测试集中，链语法解析器唯一解析完全失败的句子是：

```
But can one, really?
```

在该测试集中，很多改进后的链语法解析器没有完全解析正确的句子，主要是因为对习语的分析错误，如：

```
It has held them at bay.
```

(这里的“at bay’并不是字面意义的“bay”’).

在该测试集中，RelEx 分析完全失败的也只有 1 句：

```
Part of it is, of course
```

其输出如下：

```
Error: No target! rel=_pobj and src=of
    _advmod(be, of_course)
    _subj(be, it)
    _quantity(it, part)
    _det(it, part)
    _advmod(part, of)
```

，

但也有一些情况，RelEx 忽略了很重要的依存关系，如：

```
The expression has become quite a cliche.
```

输出中只有下列依存关系：

```
_obj(become, cliche)
 _subj(become, expression)
```

，

同时还有下列属性：

```
pos(quite_a, det)
idiom-FLAG(quite_a, T)
```

，

该例中，RelEx 忽略了“quite a” 和”cliche’之间的依存关系;

类似，在该测试集上，RelEx2Logic 也几乎都给出合理的语义关系，但有时的输出也不完整。如：

```
But there is also compassion.
```

，

RelEx2Logic 没有产生与”but“相关的语义关系，这是因为有关“but”的 RelEx2Logic 规则还没写入规则库 it does not produce a relation for “but” because at this time no “but” rule has been coded. Similarly in

总的来说，在该简单的测试集上的结果是鼓舞人心的，它能说明，本章阐述的自然语言系统在简单句子上的表现性能是不错的，其中大部分错误也都能

通过添加相关的规则来解决。当然，基于规则的系统的本质也在于能通过不断添加和修改规则来提高系统的性能，但总归是很耗时的，而且也很不灵活，因此，我们希望能使机器自动从无标注或者半标注的语料库中学习相关的语法规则来替换我们所使用的人工编写的规则库。本文第**??**章也提出了一个无监督的语言学习的思路。

5.7 本章小结

本章主要介绍我们的自然语言理解系统中的各个模块的工作原理和实现方法，并深入讨论了该系统如何将英文句子转换成超图表示的逻辑形式，以达到能使自然语言参与到逻辑推理的过程中。该自然语言理解系统已经被应用在游戏角色控制和人形机器人等领域。

该自然语言理解系统使用了一些现有的开源语法分析器，本章也讨论对这些开源工具做出的各种改进，还讨论了我们实现的全新的将依存关系转换成语义逻辑表示的系统 RelEx2Logic，总的来说，本章的贡献有：

- 改进了链语法分析器的词典，使其能够处理更复杂的语言现象，以及能识别句子的中心词从而改进了链语法对复句的分析能力。
- 改进了 RelEx 使其能处理目前语法分析器难以处理的比较级和量词等复杂的语言现象。
- 设计并实现了 RelEx2Logic 系统，使其能将 RelEx 输出的依存关系转换成以超图表示的逻辑形式
- 设计并实现了 RelEx2Logic 中所使用的转换规则及相应的规则引擎。

总的来说，本章搭建的自然语言理解系统已经能被用在一些用户定义的实际应用中，但仍然有很多遗留问题有待解决。在接下来的工作中，我们会在我们的自然语言理解系统中引入概率逻辑推理引擎 PLN，这样不仅可以通过一些基本常识推理来改进语法分析中的结果，还可用于指导对语法分析器产生的多种结果进行排序，以及用于指导各种歧义消除和指代消解等方面。除此

之外，我们还致力通过无监督的机器学习方法从大型文本语料学习相关的语法规则，来替代当前系统中使用的人工编写的规则库，本文会在第**??**章中对此做更深入的讨论。最后，本章还在一个小的测试集上对我们的自然语言理解系统进行了测试并做了相应的评价分析。

第六章　自然语言生成：从逻辑到语言

与第5章阐述的自然语言理解系统的工作（将英语句子转换成超图表示的逻辑形式）正好相反，本章将要的阐述的自然语言生成系统的工作是将基于超图表示的逻辑结构转换成合法的英语句子。本章的研究工作主要针对第1章中的提到的假设 2 展开。根据该假设，我们认为，借助一个超图转换系统和一个超图匹配系统，在由自然语言理解系统自动生成的二元组（语言表达式，逻辑表达式）组成的知识库中根据逻辑表达式找到匹配的语言表达式并生成自然语言，是可行的。

基于这样的假设，我们在自然语言生成系统中设计并实现了一个超图转换系统，用于自然语言生成中的微观规划（micro planning）；和一个超图匹配系统，用于自然语言生成中的表层生成（surface realization）。我下面章节将分别对这两个子系统的工作原理和实现方法等做进一步地讨论。

6.1　微观规划器

本章节主要介绍我们的自然语言生成系统中的用于微观规划的子系统，我们称其为微观规划器（以下称 Microplanner）。

从我们的要实现长期目标即智能会话系统的设计角度来看，对话管理模块输出的基于超图表示的逻辑表示，这些逻辑表示包含了智能会话系统想要表达的内容，即“说话内容”，说话内容的逻辑表示只是对话管理模块在没有相关的语法约束，根据当时的语境从知识库 Atomspace 中收集到的有关信息，因此，这些表示”说话内容“的超图表示可能无法直接通过在（语言表达式，逻辑表达式）的二元组的知识库中找到相应的语言表达式来生成句子。因此，我们引入了 Microplanner。Microplanner 的任务就是接收从对话系统传来的表示”说话内容“的超图表示，通过超图转换，生成能够符合表达要求的超图。

从实现角度来看，Microplanner 的工作在 OpenCog 的统一知识库 Atomspace

中进行，主要任务是将一组 Atoms（用**S**表示）和一些用于表示话语类型（如祈使句、疑问句等）的参数（该参数可以从言语行为规划器中获得，第4.3节中进一步讨论），在不改变这些 S 中所包含的”说话内容“的前提下，转换成符合输入参数所需的话语类型的，且符合英语语法的一组新的 Atoms（用**S'**表示）。需要说明的是，为了能规划出一个完整的句子，Microplanner 也有可能使用 S 之外的 Atom。在实现过程中，我们利用 Atomspace 中 SetLink（参见第节中的 Atomspace 的基本边和节点）来连接 S 中的所有 Atoms。

目前，我们的 Microplanner 还不够成熟，但已经实现了基本的功能。其中包括以下话语类型的表达:

- **陈述句（Declarative）：** 对于陈述句话语类型，Microplanner 的目标就是能表明 S 中的所含的内容且符合语法的超图表示 S ‘
- **疑问句（Interrogative）：** 对于疑问句话语类型，Microplanner 的目标可以是下面两项:
  - 对一般疑问句，征求某一个超图的边 L 的真值(L 包含 S 中的内容)
  - 对特殊疑问句，征求可以填充某个变量节点 VariableNode $V 的内容（S 构成$V 的约束）
- **祈使句（Imperative）：** 对于祈使句话语类型，其目标是产生一组 Atoms，使其包含 S 中希望听话人能服从的命令内容。
- **感叹句（Interjective）：** 对于感叹句话语类型，其目标就是能表达 S 中的相关语用内容

Microplanner 的相应算法流程可表示如下:

```
MakeUtterance(Atom-Set S, utterance type t)

\\ the utterance type t = declarative, interrogative, imperative, etc.
\\ 如果话语类型输入为空，那么系统会考虑所有的类型。

   Initialize S_leftover=S
```

```
Repeat until S_leftover = empty  :

   Make a sentence by calling MakeSentence(S_leftover, S, t)

   Let S_used = the Atoms from S used inside the
   above invocation of MakeSentence (i.e. the Atoms expressed
   in the sentence that MakeSentence produces)

   Let S_leftover = S_leftover - S_used
```

其中，针对陈述句类型的 MakeSentence 流程可表示如下（其他话语类型类似）：

```
MakeSentence(Atom-Set S_available, S_just_said, utterance type t)

   S_available = the Atoms available for utterance, within S

   S_just_said = the Atoms just said within the same utterance,
              useful for inserting anaphora

   Initialize S_leftover=S_available

  Pick a top-level sentence form (e.g. for declarative it could be SV or SVO).

   Pick a set S_start consisting of Atoms in S_leftover that match the
      top-level sentence form chosen
     (i.e. that match the output of the RelEx2Logic rule corresponding to the
      top-level sentence form)

   Let S_working = S_start

  If S_working contains Atoms from S_just_said, or from previous versions of
    S_working within this invocation of MakeSentence, then consider inserting
      anaphora to refer to them

  (**) Use SuReal (and a reverse of the chosen RelEx2Logic rule) to produce a
```

```
      sentence corresponding to S_working

   If the sentence produced is too long or complex, DONE
       (i.e., decide that the sentence is done and additional Atoms must
       go into a new sentence, referring back to the current one)

   Now, pick a RelEx2Logic rule whose output matches some set S_new of Atom-
s that
      overlaps with S_working

   Let S_working = S_working + S_new

   Goto step **

   DONE
```

目前，Microplanner 的工作主要通过两个模块来完成：组块和指代的引入。下面将具体介绍这两个模块的实现方法。

6.1.1 组块

从上面的 Microplanner 算法的伪代码可得知，Microplanner 首先选择输入 S 中的子集来进行组块，然后不断迭代，直到输入 S 中所有内容都被规划。组块的算法可描述如下：

(a) 对输入的超图（S 的子集）中的每条边根据以下几个权重因子来排序：

- form-weight (0..1): 该边是否能满足基本句式（Microplanner 根据话语类型定义了相应的基本句式）
- time-weight (n..0): 集合中的时间顺序（很多表达输出需要考虑时间顺序，比如“我要去厦门参加论文答辩”中“去厦门”和“参加论文答辩”必须按照一定顺序输入）
- link-weight (n): 当前输入的超图中与已经规划过的内容中相同节点的个数

其中默认的权值计算公式如下：$form-weight*(time-weight+link-weight)$

(b) 选择权值最高的边，调用 SuReal（在下一节中会讨论），判断该边是否是”可表达的“(即能在知识库中找到相应的语言表达式)，或者是否是“很复杂的”(即该组块能在知识库中找到超过 n 种话语类型的语言表达式，n 默认为 3)。其中“很复杂的”因素只是针对输入时没有指定话语类型的情况。

i. 如果该组块是“可表达的”，但是还能更长，那么进行以下操作：

- 对剩下的超图中的所有边重新排序，但这次排序的目的在于选择不同于已使用的句式的边 A。这样考虑是因为，如果已使用的句式不能完全表达 A 中的内容，那么可以试试其他的句式。
  - form-weight (0..1): 同上
  - time-weight (n..0): 同上
  - link-weight (n): 当前的超图中与即将说的内容中的相同节点的个数

  其中默认的权值计算公式如下：$(time-weights+link-weights)*(2-form-weights)$
- 选择权值最高的边，然后继续调用 SuReal 试着再说一遍。

ii. 如果该组块不是“可表达的”，那么：

- 检查当前组块，看是否有节点只出现过 1 次
- 在该节点对应的边加入组块中
- 继续这样尝试 n 次（n 默认为 3）后，如果该组块还不是“可表达的”那么放弃。

6.1.2　指代的引入

为了让句子的表达更具语言色彩以及更符合人类的交流习惯，还需要对已经描述的对象进行指代处理，也就是利用代词、固定名、完整或缩略名词短语

来替换那些已经被描述过的对象，因此 Microplanner 设计了引入指代处理的模块。其实现步骤可以表示如下：

(a) 遍历所有的组块，找到所有的名词，并生成一个名词序列。

(b) 对名词序列中的每一个名词，通过查询知识库中与其相关的 RelEx 关系，选定能在替换该名词时所要使用的代词的根，这里的根指"he", "she", "it", "they"等。例如，“Mary”通常是表示女名，那么这里就指定能替换”Mary“的代词的根”she“。

(c) 遍历每个名词，判断它是否可以被替换成代词：

- 如果该名词从来没有被提及过，或者在很久前被提及过（超过 n 个组块的范围，n 默认为 3），那么该名词不能被替换成代词。
- 如果该名词在同一个组块里被 InheritanceLink 修饰，那么该名词不能被替换成代词。(英文中形容词后面的名词不能被替换成代词)
- 检查该名词在名词序列中的前 3 个词，如果其中一个已经被替换成代词了，那么为了降低表达的歧义，这里也认为该名词不能被替换成代词。

(d) 根据上面的判断结果，对可以被替换成代词的名词进行如下代词替换操作：

- 如果名词是主语，将代词保持主格形式，(即"I", "he", "they"等)
- 如果名词在被赋值为“所有格”的 EvaluationLink 中，那么将其改为所有格形式（"my" "his" "its"等）
- 如果名词是宾语或者间接宾语
  - 如果它和主语是相同名词，那么根据情况替换成"myself", "himself", "themselves"等。
  - 如果它和主语不同，那么根据情况替换成"me", "him", "them"等

(e) 如果一个名词无法被换成代词，将使用下列算法来进一步考虑它是否可以被相关的名词替代（例如“松树”可以用“树”取代）：

- 通过 InheritanceLink 查找与该名词节点有继承关系的名词节点

- 根据下面公式对这些有继承关系的名词进行打分：(□□□□□□□□□□□□□□) ∗ (□□□*InheritanceLink*□□□*Strength*)∗(□□*InheritanceLink*□□□□*Confidence*)
- 选择得分最高的名词去替换（如果最高得分的名词不止一个，那么随机选择一个）

6.1.3　存在问题和改进方向

我们这里讨论的 Microplanner 虽然能处理一般的现象，并且能和后面章节要讨论的对话管理以及表层生成等模块结合起来用于简单的对话系统。但还有很多方面需要改进。一般来说，微观规划过程还包含选词处理，但由于我们目前使用的知识库还比较小，可以直接通过 InheritanceLink 找到相关词，所以这一模块暂时被搁浅，我们会在下一个版本中实现选词模块以及加入更多话语类型的处理模块。除此之外，我们还可以在微观规划过程中引入简单的逻辑理论，进一步改进指代的引入。

6.2　表层生成器 SuReal

表层生成是将经微观规划后的内容描述映射至由文字、标点符号和结构注解信息组成的表层文本的过程。在第**??**中提到，表层生成的算法有很多种，而且由于本身“表层生成”的概念没有明确的定义，不同的表层生成系统的输入都不同，很难有可比性。本文从人脑或类人脑系统的生成语言的思路出发，结合机器具备大型计算能力的优点，设计并在 OpenCog 中的知识库上实现了基于超图近似匹配的表层生成器（**Su**rface **Real**izer，以下简称 SuReal），使其能将微观规划后的超图转换成英文句子。本节将会具体阐述其工作原理和实现方法。

6.2.1　SuReal 算法

概括地说，SuReal 的工作可以看成是第5中描述的自然语言理解系统 Link Parser, RelEx, RelEx2Logic 的逆过程，但是，实现这样的逆转换并没那么简

单，因为在将英文句子转成逻辑形式的过程中抽象了很多词法或语法关系等重要语言表层现象。而且在我们的自然语言系统中创建的一系列规则也只是同态而并不同构，无法通过简单地反向使用同样的规则集来实现这个逆转过程。

基于这些考虑，我们提出了一个全新的表层生成方法，它以基于超图的逻辑表达式作为输入，但不是直接在语义层面上将此逻辑表达式与知识库中的其他逻辑表达式进行匹配，而是将句法分析结果也转换成超图形式引入到知识库中，然后在句法规则的指导下进行基于超图的匹配。SuReal 的实现原理操作流程如下：

- 使用第5章中的自然语言理解系统分析一系列英文句子，然后将分析后得到的语义结构以及中间步骤得到的句法结构形成配对的二元组，以（语言表达式，句法表达式 + 逻辑表达式）的形式存入到知识库 Atomspace 中。其中逻辑表达式是指 RelEx2Logic 产生的使用超图表示的逻辑形式，句法表达式是链语法分析产生的结果并存入到 Atomspace 中的超图形式；语言表达式即以超图中的 SentenceNode 节点表示的英文句子。句法表达式在 Atomspace 中的超图表示如下：

```
;链语法分析结构中”the”和“cat”之间的链接的超图表示

(EvaluationLink (stv 1.0 1.0)
   (LgLinkInstanceNode "Ds**c@4432ef3-3c4e-42a9-8072-9975d168a12c")
   (ListLink
      (WordInstanceNode "the@da65d87c-22b9-4af2-89f4-60042816c579")
      (WordInstanceNode "cat@1a6d58eb-e9c0-4c8e-af01-8d4304e3430c")
   )
)

;链语法分析结构中链Ds**c的超图表示

(LgLinkInstanceLink
   (LgLinkInstanceNode "Ds**c@4432ef3-3c4e-42a9-8072-9975d168a12c")
   (LgConnector
      (LgConnectorNode "D")
```

```
        (LgConnDirNode "+")
    )
    (LgConnector
        (LgConnectorNode "Ds**c")
        (LgConnDirNode "-")
    )
)
```

- SuReal 的输入需要用 SetLink 连接用于生成句子的逻辑表达式，这里假设输入为 SetLink S，那么可通过以下操作实现 SuReal(S)：
  (a) 对给定的 SetLink S 中的某个节点元素，在链语法词典中查找对应的链语法链接要求，并以 DNF(Disjunctive Normal Form)的形式存入 Atomspace 中
  (b) 调用 OpenCog 中的模式匹配器（Pattern Matcher），将 S 中的每个节点携带相应链语法链接要求的词节点当成变量，与预先存入知识库中的二元组中的逻辑表达式和句法表达式进行匹配。
  (c) 当模式匹配器在知识库中搜索到对应的匹配后，检查其中的词节点携带的链语法链接要求是否与 S 中对应节点携带的链语法链接要求 DNF 是否吻合，如果含有相似的链接要求，则将匹配到的超图中的词节点替换成 S 中对应的词节点。
  (d) 将匹配结果在二元组中对应的语言表达式中的相应词替换成新的词节点对应的词后，生成句子。
  (e) 对上一步得到的所有句子进行排序后输出最终结果。排序中考虑的因素有语言模型、已经原有句子中被替换掉的词的个数等。

6.2.2　综观 SuReal 及其存在的问题和改进方向

XXXX 需要翻译

Very broadly speaking, what this algorithm is doing is simply to reversing the various graph and hypergraph rewrite rules described in the previous sections on the

Link Parser, RelEx and RelEx2Logic. However, this is not entirely simple, because the rules create homomorphisms rather than isomorphisms. Any one Atom structure may be produced by many different link-grammar structures, because there are many grammatical ways to produce any given idea. But not all the grammatical structures corresponding to different subsets of a given Atom set needing articulation, will necessarily be grammatically compatible with each other. So one has a constraint satisfaction problem, which in general will have multiple solutions, with varying levels of syntactic ambiguity and subjective human naturalness. The above algorithm represents a heuristic approach to this problem.

More precisely, suppose we have an Atom set $A = \{A_i\}$; and let $R = \{R_j\}$ denote the set of all graph rewrite rules R_j with the property that R_j maps at least one link parse subtree into some nonempty subset of $\{A_i\}$. Let $R^i \in R$ denote the set of rewrite rules that produce an Atom set including the particular Atom A_i; we may write $R^i = \{R^i_k\}$, with $R = \bigcup_i R^i$. Let $m_g(r)$ denote the proposition that the rewrite rule r matches some subgraph of the graph g.

Given this set-up, the problem of generating a sentence expressing the Atom set A boils down to finding some link parse g that

- parses correctly according to link grammar
- satisfies the expressiveness condition

$$\bigwedge_i \bigvee_k m_g(R^i_k),$$

- satisfies an assumed "aesthetic condition", initially: that it would either not parse or not satisfy the expressiveness condition if any of its words were removed

Given a link parse g, producing the relevant sentence is trivial. The task of generating a sentence-set expressing A reduces to choosing a way to partition A into subsets, so that each can be acceptably expressed via a single sentence.

A strength and weakness of this approach is that, in most practical cases, this constraint satisfaction problem will have many solutions. Selection among the various solutions could be approached in many ways, e.g. via evaluating various solutions and choosing the one with the highest word tuple probabilities relative to a large reference corpus; or via proceeding as in the current NLGen system, and choosing solutions whose fragments are known via past NL comprehension experience to have been used in real human-generated sentences.

Another strength and weakness of this approach is that it is data-driven. The system can only generate sentences whose fragments resemble previously-encountered sentences. For this reason, from an AGI perspective, this approach can ultimately serve only as a portion of an overall language generation system. It must be augmented by a method with less limited creative power, even if this additional method is slower. But for medium-term practical purposes, there are many more critical challenges than expanding the creative potential of the surface realization component of our system; the current approach seems more than adequate for the time being.

目前 SuReal 存在的问题有：

- 受限于 RelEx2Logic 的输出。由于匹配所使用的三元组知识库是通过我们的自然语言理解系统产生的，那么其中的逻辑表达式中的所有 Atoms 类型只能来源于 RelEx2Logic 中定义的，因此如果输入中含有 RelEx2Logic 中未使用过的 Atom 类型，那么将无法生成句子。
- 尚未将 OpenCog 中的强化学习使用到匹配中，使匹配更智能简洁。

可以改进方向包括：

- 使用强化学习来实现更智能更灵活的超图匹配。
- 改进 RelEx2Logic 使其能提供更丰富的逻辑关系，从而使通过我们的自然语言理解系统分析并存入知识库的查询库更丰富。

6.2.3 实验结果示例及简要分析

正如前面提到，不同的自然语言生成系统所使用的输入形式都不一样，有词语层次的输入，也有句法和语义层次的输入，导致研究的方向不一样，算法也千差万别，因此目前并没有权威的评估标准。我们这里只给出本系统的一些实验结果并简要分析。

首先给出一个句子集如下：

```
Play a song by Weird Al Yankovic.
Play another song by Weird Al.
Play something by The Cure.
Who wrote 'Blue Monday'?
What is the best song by New Order?
When did 'Thriller' come out?
I want to hear some 60's soul music.
Can you play something new for me?
He bought a guitar in the store.
Madonna sang the song called 'Vogue'.
```

然后运行我们的自然语言处理系统，将得到的相应的（语言表达式，逻辑表达式）二元组导入 OpenCog 的知识库 Atomspace 中。运行 SuReal 可以得到如下实验结果：

输入：

```
(SetLink
    (EvaluationLink
        (PredicateNode "ate")
        (ListLink
            (ConceptNode "John")
            (ConceptNode "pig")
        )
    )
)
```

输出：

```
(John ate a pig.)
(John ate the pig.)
```

不难看出，第一个输出(Jonh ate a pig.)是匹配导入的句子集合中“He bought a guitar in the store.”对应的超图的子图成功后得到的结果；第二个输出(John ate the pig.)是匹配其中“Madonna sang the song called 'Vogue'.”对应的超图的子图成功后得到的结果。

输入：

```
    (SetLink
        (EvaluationLink
            (PredicateNode "sang")
            (ListLink (VariableNode "$ABC"))
        )
    )
```

输出：

```
(who sang ['] Blue Monday ['] ?)
```

该输入中含有一个变量，说明是由 Microplanner 规划的疑问句形式，而且根据输入的逻辑表达式可以看出，该变量充当主语的角色。因此可以通过匹配“Who wrote 'Blue Monday'?”后得到相应的结果。

6.3　本章小节

本章针对自然语言生成，设计并实现了一个新的自然语言生成系统，能将基于超图表示的逻辑形式转换成英语句子形式。由于使用独特的知识表示系统，该设计完全是本文特有的构想。当然，该语言生成系统也有一定的局限性，比如 Microplanner 目前只能处理一定的话语类型，SuReal 无法生成知识库中不存在的表达。但我们的实验结果表明，它们已经能完全运用在简单的句子生成上，有时候甚至也能处理稍微复杂一点的情况。

第七章　基于超图表示的语言逻辑推理的设计与实现

基于自然语言的逻辑推理在很多应用领域都起到了非常重要的作用，其中也包括本文第**??**章将介绍的智能会话系统。针对第**??**章阐述的基于超图的知识表示以及逻辑推理系统等的理论知识，并在第5章介绍的自然语言理解系统的基础上，我们进一步设计并实现了一个基于超图表示的语言逻辑推理系统。本章将通过几个典型例子来介绍该语言逻辑推理系统的设计思路和实现步骤，也就是将自然语言通过自然语言理解系统转换成超图的表示形式，并使用概率逻辑网 PLN 来进行一些常识推理的过程。

7.0.1　有关比较级的推理

首先，紧接着上一章中对比较级的句法分析的讨论，我们这里通过几个例子来展示，在我们的自然语言系统输出的有关比较级的逻辑表达式上，PLN 如何实现相关推理。为了更好地显示有效的推理过程，我们在下面的例子中过滤了和比较级推理无关的 RelEx 关系和逻辑表达。

假设有：

- Bob likes Hendrix more than the Beatles
- Bob is American
- Menudo is liked less by Americans than the Beatles

那么我们可以推出：Bob likes Hendrix more than Menudo.

对于第一个句子，通过执行 RelEx 和 RelEx2Logic，可以得到：

```
_subj(like, Bob)
_obj(like, Hendrix)
than(Hendrix, Beatles)
_comparative(like, Hendrix)
==>
```

```
TruthValueGreaterThanLink
    EvaluationLink like Bob Hendrix
    EvaluationLink like Bob Beatles
```

类似地，第二个句子：

```
_subj(like, Americans)
_obj(like, Menudo)
than(Beatles, Menudo)
_comparative(like, Beatles)
==>
TruthValueGreaterThanLink
    EvaluationLink like Americans Beatles
    EvaluationLink like Americans Menudo
```

从这些句子中得到的逻辑表达是非常透明的。简单地根据“TruthValueGreaterThan”关系是可传播，以及“Bob is American”，PLN 很容易推理得到结论：

```
TruthValueGreaterThanLink
    EvaluationLink like Bob Hendrix
    EvaluationLink like Bob Menudo
```

那么现在所处理的知识是逻辑形式而非句法形式，因此可以调用知识库中的相关知识加入到推理过程中。例如，假设我们还知道“Bob likes Sinatra more than Menudo”，可表示如下：

```
TruthValueGreaterThanLink
    EvaluationLink like Bob Hendrix
    EvaluationLink like Bob Menudo
```

根据 PLN 中的回溯推理规则 $((A \to C) \wedge (B \to C) \Rightarrow (A \to C))$,可以得到：

```
SimilarityLink
    Hendrix
    Sinatra
```

7.0.2　基于自然语言的三段论推理

为了测试 RelEx2Logic 的输出用于逻辑推理的有效性，我们使用了 PLN 来执

行基于自然语言输入的三段论的推理。接下来，我们给出一个基于自然语言的三段论推理的例子，和详细的推理步骤。

假设我们需要完成的推理如下:

```
Socrates is a man.（苏格拉底是人）
Men breath air.（人呼吸空气）
|-
Socrates breath air.（苏格拉底呼吸空气）
```

对于此例，推理前提在 Atomspace 中可表示如下:

```
Concepts

(ConceptNode "Socrates@f250f429-5078-4453-8662-c63cc8f58a22") ; [217]

(ConceptNode "Socrates" (stv .1 1.0)) ; [218]

(ConceptNode "man@80d4e852-1b93-4c49-84e1-5a7352b0dcb1" (stv .1 1.0)) ; [220]

(ConceptNode "men@83d83a2e-940c-46aa-91f1-a7d2c106ef13" (stv .1 1.0)) ; [290]

(ConceptNode "man" (stv .1 1.0)) ; [221]

(ConceptNode "air@e3f175ea-c3d1-45cf-883d-a6d2f6a879ac") ; [292]

(ConceptNode "air") ; [293]

(ConceptNode "present") ; [227]满足

breathe(x,y的多元组可表示如下: )

(ListLink (stv 1.000000 0.000000)
  (ConceptNode "men@83d83a2e-940c-46aa-91f1-a7d2c106ef13") ; [290]
  (ConceptNode "air@e3f175ea-c3d1-45cf-883d-a6d2f6a879ac") ; [292]
) ; [295]

breathe(x,y)
```

```
(EvaluationLink (stv 1.000000 1.000000)
  (PredicateNode "breathe@218b15b2-52a8-430c-93f6-b4bba225418c") ; [287]
  (ListLink (stv 1.000000 0.000000)
    (ConceptNode "men@83d83a2e-940c-46aa-91f1-a7d2c106ef13") ; [290]
    (ConceptNode "air@e3f175ea-c3d1-45cf-883d-a6d2f6a879ac") ; [292]
  ) ; [295]
) ; [296]

"Socrates IS-A man（“苏格拉底是人”）可表示如下："

(InheritanceLink (stv 1.000000 1.000000)
  (ConceptNode "Socrates@f250f429-5078-4453-8662-c63cc8f58a22") ; [217]
  (ConceptNode "man@80d4e852-1b93-4c49-84e1-5a7352b0dcb1") ; [220]
) ; [223]具体实例继承于泛化概念：



(InheritanceLink (stv 1.000000 0.990000)
  (ConceptNode "men@83d83a2e-940c-46aa-91f1-a7d2c106ef13") ; [290]
  (ConceptNode "man") ; [221]
) ; [291]

(InheritanceLink (stv 1.000000 1.000000)
  (ConceptNode "Socrates@f250f429-5078-4453-8662-c63cc8f58a22") ; [217]
  (ConceptNode "Socrates") ; [218]
) ; [219]

(InheritanceLink (stv 1.000000 0.990000)
  (ConceptNode "man@80d4e852-1b93-4c49-84e1-5a7352b0dcb1") ; [220]
  (ConceptNode "man") ; [221]
) ; [222]
```

下图显示了 PLN 的推理路径以及每一步中使用的推理规则：

1）成员变量泛化规则（）GeneralEvaluationToMemberRule输入：

```
(EvaluationLink (stv 1.000000 1.000000)
  (PredicateNode "breathe@7f5b37e8-e4b3-4335-a06b-68af470cf354") ; [350]
  (ListLink (stv 1.000000 0.000000)
```

```
    (ConceptNode "men@a2905bdd-9214-4717-82c6-dfe21c1263bc") ; [353]
    (ConceptNode "air@9bfe790d-5446-4219-be74-845c0d145fe1") ; [355]
  ) ; [358]
) ; [359]输出：以及

(中其他变量被泛化后的个其他变体) ListLink3
(MemberLink (stv 1.000000 1.000000)
  (ConceptNode "men@a2905bdd-9214-4717-82c6-dfe21c1263bc") ; [353]
  (SatisfyingSetLink (stv 1.000000 1.000000)
    (VariableNode "$X0") ; [441]
    (EvaluationLink (stv 1.000000 0.000000)
      (PredicateNode "breathe@7f5b37e8-e4b3-4335-a06b-68af470cf354") ; [350]
      (ListLink (stv 1.000000 0.000000)
        (VariableNode "$X0") ; [441]
        (ConceptNode "air@9bfe790d-5446-4219-be74-845c0d145fe1") ; [355]
      ) ; [443]
    ) ; [444]
  ) ; [445]
) ; [446]
```

2) 成员继承规则（MemberToInheritance ）Rule输入：上一步的输出输出：

```
(InheritanceLink (stv 1.000000 1.000000)
  (ConceptNode "men@a2905bdd-9214-4717-82c6-dfe21c1263bc") ; [353]
  (SatisfyingSetLink (stv 1.000000 1.000000)
    (VariableNode "$X0") ; [441]
    (EvaluationLink (stv 1.000000 0.000000)
      (PredicateNode "breathe@7f5b37e8-e4b3-4335-a06b-68af470cf354") ; [350]
      (ListLink (stv 1.000000 0.000000)
        (VariableNode "$X0") ; [441]
        (ConceptNode "air@9bfe790d-5446-4219-be74-845c0d145fe1") ; [355]
      ) ; [443]
    ) ; [444]
  ) ; [445]
) ; [6107]
```

3) 溯因推理规则 () AbductionRule输入:

```
(InheritanceLink (stv 1.000000 0.990000)
  (ConceptNode "men@a2905bdd-9214-4717-82c6-dfe21c1263bc") ; [353]
  (ConceptNode "man") ; [284]
) ; [354]

(InheritanceLink (stv 1.000000 0.990000)
  (ConceptNode "man@fbc51aff-8074-46d8-b3ba-1ccaeb1adb34") ; [283]
  (ConceptNode "man") ; [284]
) ; [285]输出:


(InheritanceLink (stv 1.000000 0.988901)
  (ConceptNode "man@fbc51aff-8074-46d8-b3ba-1ccaeb1adb34") ; [283]
  (ConceptNode "men@a2905bdd-9214-4717-82c6-dfe21c1263bc") ; [353]
) ; [609]
```

4) 演绎推理规则 () DeductionRule输入: 上一条输出以及下面的原子集合

```
(InheritanceLink (stv 1.000000 1.000000)
  (ConceptNode "Socrates@46ec3d0f-4535-4d01-87b7-84ef65c25a23") ; [280]
  (ConceptNode "man@fbc51aff-8074-46d8-b3ba-1ccaeb1adb34") ; [283]
) ; [286]输出:


(InheritanceLink (stv 1.000000 0.991755)
  (ConceptNode "Socrates@46ec3d0f-4535-4d01-87b7-84ef65c25a23") ; [280]
  (ConceptNode "men@a2905bdd-9214-4717-82c6-dfe21c1263bc") ; [353]
) ; [777]
```

5) 演绎推理规则 () DeductionRule输入: 步骤) 和步骤) 的输出
24输出:

```
(InheritanceLink (stv 1.000000 0.986333)
  (ConceptNode "Socrates@46ec3d0f-4535-4d01-87b7-84ef65c25a23") ; [280]
  (SatisfyingSetLink (stv 1.000000 0.000000)
```

```
    (VariableNode "$X1") ; [442]
    (EvaluationLink (stv 1.000000 1.000000)
      (PredicateNode "breathe@7f5b37e8-e4b3-4335-a06b-68af470cf354") ; [350]
      (ListLink (stv 1.000000 0.000000)
        (VariableNode "$X1") ; [442]
        (ConceptNode "air@9bfe790d-5446-4219-be74-845c0d145fe1") ; [355]
      ) ; [922]
    ) ; [923]
  ) ; [8605]
) ; [12317]
```

6) 成员变量继承规则（）InheritanceToMemberRule输入：上一条规则的输出输出：

```
(MemberLink (stv 1.000000 0.989841)
  (ConceptNode "Socrates@46ec3d0f-4535-4d01-87b7-84ef65c25a23") ; [217]
  (SatisfyingSetLink (stv 1.000000 1.000000)
    (VariableNode "$X0") ; [385]
    (EvaluationLink (stv 1.000000 0.000000)
      (PredicateNode "breathe@7f5b37e8-e4b3-4335-a06b-68af470cf354") ; [350]
      (ListLink (stv 1.000000 0.000000)
        (VariableNode "$X1") ; [442]
        (ConceptNode "air@9bfe790d-5446-4219-be74-845c0d145fe1") ; [355]
      ) ; [922]
    ) ; [388]
  ) ; [389]
) ; [6375]
```

7) 成员变量赋值规则（）MemberToEvaluationRule输入：上一条规则的输出最终推理结论：

```
(EvaluationLink (stv 1.000000 0.989841)
  (PredicateNode "breathe@7f5b37e8-e4b3-4335-a06b-68af470cf354") ; [350]
  (ListLink (stv 1.000000 0.000000)
    (ConceptNode "Socrates@46ec3d0f-4535-4d01-87b7-84ef65c25a23") ; [217]
```

```
    (ConceptNode "air@9bfe790d-5446-4219-be74-845c0d145fe1") ; [355]
  ) ; [922]
) ; [388]
```

需要说明的是，上述提供的从前提到结论的推理路径中，只显示了在推理过程中提供有用信息的推理步骤。在推理过程中，PLN 的后向推理链尝试并放弃大量其他的可能推理步骤。

7.1 本章小结

本章介绍了几个使用自然语言理解系统的输出在 PLN 上的推理实验。实验表明，我们的自然语言理解系统输出的逻辑形式能和逻辑推理引擎结合，并能进行简单的常识推理。但是目前的系统还无法完成较复杂的推理。我们下一步将继续改进 RelEx2Logic 系统，使其能更好地为复杂的句子提供更准确的逻辑表达，并改善 PLN 的控制机制，使其能进行更复杂的推理。

第八章　基于概率逻辑推理的问答系统的设计和实现

XXXX

8.1　基于超图模糊匹配的问答系统

为了实现我们的智能对话系统的框架以及设计思路，我们构建了一个基于超图模糊匹配的问答系统的原型，该问答系统使用前文提到的自然语言理解技术将自然语言转换成以超图表示的语义逻辑形式，利用基于超图匹配的动态编程算法，从而在问答语料库中找到最适合问题的答案来响应用户查询。

本节首先讨论该问答系统框架中使用的超图匹配算法，然后再讨论基于超图模糊匹配的问答系统的设计和实现。

8.1.1　基于超图的模式匹配器

基于超图的模式匹配器是一个查询引擎或变数配对者，主要功用是在我们的超图知识库 Atomspace 中寻找特定的模式，或是一些 Atom 相关排列或“模板“。在轮入一个等定的 Atom 排列（即一个超图）后，模式匹配器会从 Atomspace 中寻找所有合乎条件的超图。同时一些“空白位置“，即“变数“在一个超图的位置，亦可存在于该输入的超图当中，而模式匹配器则会“填补“这些“空白位置“。例如：“John threw a ____). “亦可以被判断为“John threw a ball. “，前提是在查询前该句子要存在于 Atomspace 里，而在这个例子中“ball“就是被配对的一个答案。输入的超图中可以在不同的地方拥有多过一个的“空白位置“，同时亦可以有多过一个的答案。在这过程中，BindLink 提供了一个便利的 API 去达到这目的，接下来会有更多的技术层面说明。

模式匹配器是一个变数配对者，是在传统计算机科学中“统一“的概念，因为这是它的功能之一。它同时亦是一个查询引擎，因为这个变数配对过程某

情度上是等同于用 SQL 在关联数据库中进行查询。其中主要的分别在于在我们这里的概念是以超图的形式表示，所以它亦可以被形容为一个超图查询语言 HQL（Hypergraph Query Language）。但这些都只是以不同的名称去表示同一个程序。

这个模式匹配器是结构精密应用的一个重要组件，同时也是建立逻辑推理系统中前向和后向链接推理的一个重要基础。它拥有 C++ 和 Scheme 的接口以供不同的应用。

模式匹配器由数个不同的组件连接在一起，组成一个基础前向链接或超图重写的工具。在其核心是一个能比较和配对不同超图以及其变数的组件，而这个组件跟另一个“找寻器“一同使用便能寻找整个 Atomspace 中的合乎条件的超图。最后一个组件是一个“编写器“，主要功用是当有一个配对成功产生后建立一个或以上在 ImplicationLink 后半部列明的超图。

这个模式匹配器接受有指定“规则“的 BindLinks，再把这些“规则“应用到 Atomspace 里。每一个“规则“是一个 ImplicationLink，以“if P then Q“的形式表达，当中 P 和 Q 分别代表不同的超图。如果 P 被实现了，那么便能得到 Q。Q 可以是任何种类的超图，但如果 Q 是一个 ExecutionLink，这便意味著当 P 被实现了，系统便将会实行其他程序，这在整体来说可以是一个十分强大的功能，因为在一般情况下很多程序都可以以 ExecutionLink 来执行。

模式匹配器不会更改任何超图中的真值（Truth Value）或关注值（Attention Value），有需要时使用者亦可以自行编写和执行相关的程序。

在默认的回调函数中模式匹配器会搜索整个 Atomspace，因此亦有需要编写特定的回调函数以限制其搜索范围，例如只寻找和接受拥有某程度短期重要性的 Atom，以只获取相对重要的资讯。

以下是一个使用模式匹配器的示例：

```
(define find-animals
  (BindLink
    ;; The variable to be bound
    (VariableNode "$var")
```

```
    (ImplicationLink
      ;; The pattern to be searched for
      (InheritanceLink
         (VariableNode "$var")
         (ConceptNode "animal")
      )

      ;; The value to be returned.
      (VariableNode "$var")
    )
  )
)
```

执行时，只需在我们系统的 Scheme 终端输入以下指令便能执行以上的模式匹配器并找出在 Atomspace 中所有继承“animal“的概念：

```
(cog-bind find-animals)
```

8.1.2　基于超图模糊匹配的问答系统

本小节我们将阐述这个基于超图的模糊匹配的启发式的问答系统的设计思路和实现方法，该系统主要针对信息查询。其基本算法可简单概括如下：给定一个查询 Q，通过我们的自然语言理解流水线将其转换成 Atomspace 里的超图形式，然后在知识库 Atomspace 里找出与其相似的表示陈述句的超图。这里我们做了一个合理的猜测，认为这些相似的超图中的其中一个包含了 Q 的答案。

如果给定一个很大的超图 H，在 H 的所有子图中找到与目标超图 Q 部分匹配的子超图是一个非常困难的计算问题。我们使用了启发式在一定程度上解决了计算难度，虽然这不能保证找到问题的正确答案，但实验发现该方法通常能找到一个很好的答案。我们的启发式涉及下面两个阶段：

- **第一阶段**：使用一个基于超图的模糊匹配器 Pattern Matcher，搜索能正确回答查询 Q 的答案。在此搜索过程中，保存所有与查询 Q 近似匹配的子超图。
- **第二阶段**：对第一阶段中的所有部分匹配的子超图，使用基于超图匹配的动态规划来计算它们的匹配程度，然后根据匹配程度进行排序。

基于超图匹配的动态规划是对[?]中的算法一个新的改进。正如基于动态规划的字符串匹配，获得高性能的一个重要因素是从源到目标的路径转换过程中合理地分配具体的操作成本。我们对此使用的启发式如下：修改（可以是添加、删除或替换）一个稀有词对应的节点比修改一个常用词对应的节点的成本高；修改一个稀有词节点对应的链比修改一个常用词节点对应的链的成本高。这里的“稀有程度”通过词频或者该词所关联的 WordNode, ConceptNode 或者 PredicateNode 的真值来衡量。

假设所需的查询是“Who bought a glockenspiel at the store?”(谁在商店买了一个钟琴？)，那么如果将“glockenspiel”替换成其他词的时候，那其成本就比将“store”替换成其他词要高，因为“glockenspiel”比“store”更稀有。因此“Bob bought a glockenspiel with his friend.”（Bob 和他朋友一起买了钟琴。）和查询的匹配程度要高于“Jim bought a thimble at the store.”（Jim 在商店买了一个针箍。）。

这种基于频率的启发式相当于实例化 OpenCog 里经常使用的一个一般原则：信息含量往往通过概率化的惊讶度来衡量。发现句子里的“glockenspiel”比发现“store”的惊讶度更高，因此，我们断定，在此查询中，“glockenspiel”含有的信息量更大，所以当修改“glockenspiel”的时候会被分配很高的成分，因为做这样的修改就相当于删除了查询中的较多的信息。这样的信息论原则也给 OpenCog 中的 Pattern Mining[?] 奠定了基础, Pattern Mining 在 OpenCog 的动机机制中充当了重要的角色，可以用来满足我们前面提到的 OpenPsi 里的“新颖性”（Novelty）的目标需求。一个更智能更复杂的匹配方法可以根据总体惊讶值来惩罚修改序列，但目前的做法依然是只针对查询的各个部分的惊讶度单独作为指标，待系统逐步改善后，会考虑实现更复杂的匹配算法。

为了简化操作，在做查询匹配的过程中，我们忽略了表示查询的超图中很多不重要的 Atoms，仅表明语义关系的核心部分用于匹配。对于例句“Who bought a glockenspiel at the store?”，仅有下面这些 Atoms 参与超图匹配：

```
((ReferenceLink
   (InterpretationNode "sentence@ae443d59-33e3-453f-b77e-2c46723f584a_parse_0_interpretation_$X")
   (SetLink
      (ImplicationLink (stv 0.99000001 0.99000001)
         (PredicateNode "bought@530d4f1a-4ad0-4b8b-b686-36f4986a0db5" (stv 0.001 0.99000001))
         (PredicateNode "buy" (ptv 0.001 0.99000001 1))
      )
      (InheritanceLink (stv 0.99000001 0.99000001)
         (ConceptNode "glockenspiel@bf4ad4a0-d48b-4c21-b817-a361757a951d" (stv 0.001 0.99000001))
         (ConceptNode "glockenspiel" (ptv 0.001 0.99000001 1))
      )
      (InheritanceLink (stv 0.99000001 0.99000001)
         (ConceptNode "store@ae4eb998-d091-45d7-8b3e-5fa1c222aab6" (stv 0.001 0.99000001))
         (ConceptNode "store" (ptv 0.001 0.99000001 1))
      )
      (EvaluationLink (stv 0.99000001 0.99000001)
         (PredicateNode "bought@530d4f1a-4ad0-4b8b-b686-36f4986a0db5" (stv 0.001 0.99000001))
         (ListLink (stv 0.99000001 0.99000001)
            (VariableNode "$rIrXGzhMvgVEaXQEaFYaXhMvzNQ5RL0m32kZ" (stv 0.001 0.99000001))
            (ConceptNode "glockenspiel@bf4ad4a0-d48b-4c21-b817-a361757a951d" (stv 0.001 0.99000001))
            (ConceptNode "store@ae4eb998-d091-45d7-8b3e-5fa1c222aab6" (stv 0.001 0.99000001))
         )
      )
      (InheritanceLink (stv 0.99000001 0.99000001)
         (ConceptNode "at@d689e641-c1f9-49a0-8e60-d2eb824f0c7a" (stv 0.001 0.99000001))
         (ConceptNode "at" (ptv 0.001 0.99000001 1))
      )
      (InheritanceLink (stv 0.99000001 0.99000001)
         (SatisfyingSetLink (stv 0.99000001 0.99000001)
            (PredicateNode "bought@530d4f1a-4ad0-4b8b-b686-36f4986a0db5" (stv 0.001 0.99000001))
         )
         (ConceptNode "at@d689e641-c1f9-49a0-8e60-d2eb824f0c7a" (stv 0.001 0.99000001))
      )
      (InheritanceLink (stv 0.99000001 0.99000001)
         (PredicateNode "bought@530d4f1a-4ad0-4b8b-b686-36f4986a0db5" (stv 0.001 0.99000001))
         (ConceptNode "past" (stv 0.001 0.99000001))
      )
      (EvaluationLink (stv 0.99000001 0.99000001)
         (PredicateNode "definite" (stv 0.001 0.99000001))
         (ListLink (stv 0.99000001 0.99000001)
            (ConceptNode "store@ae4eb998-d091-45d7-8b3e-5fa1c222aab6" (stv 0.001 0.99000001))
         )
      )
      (ImplicationLink (stv 0.99000001 0.99000001)
         (PredicateNode "at@d689e641-c1f9-49a0-8e60-d2eb824f0c7a" (stv 0.001 0.99000001))
         (PredicateNode "at" (ptv 0.001 0.99000001 1))
      )
      (EvaluationLink (stv 0.99000001 0.99000001)
         (PredicateNode "at@d689e641-c1f9-49a0-8e60-d2eb824f0c7a" (stv 0.001 0.99000001))
         (ListLink (stv 0.99000001 0.99000001)
            (ConceptNode "store@ae4eb998-d091-45d7-8b3e-5fa1c222aab6" (stv 0.001 0.99000001))
         )
      )
      (InheritanceLink (stv 0.001 0.99000001)
         (InterpretationNode "sentence@ae443d59-33e3-453f-b77e-2c46723f584a_parse_0_interpretation_$X")
         (ConceptNode "InterrogativeSpeechAct" (stv 0.001 0.99000001))
      )
   )
)
)
```

类似地，我们使用简化过的表示“Bob mauled a glockenspiel with his friend.”的超图：

```
((ReferenceLink
   (InterpretationNode "sentence@47794171-b923-41e5-a03a-7fc22eb583fb_parse_0_interpretation_$X")
   (SetLink
      (ImplicationLink (stv 0.99000001 0.99000001)
         (PredicateNode "mauled@1a06458b-e649-431c-9430-5940e43bbf21" (stv 0.001 0.99000001))
         (PredicateNode "maul" (ptv 0.001 0.99000001 1))
      )
      (InheritanceLink (stv 0.99000001 0.99000001)
         (ConceptNode "Bob@1c332525-f7a2-4b72-9256-9e32f0d5e9da" (stv 0.001 0.99000001))
         (ConceptNode "Bob" (ptv 0.001 0.99000001 1))
      )
      (InheritanceLink (stv 0.99000001 0.99000001)
         (ConceptNode "glockenspiel@b1288105-63bc-4630-bf0b-e35a585647ee" (stv 0.001 0.99000001))
         (ConceptNode "glockenspiel" (ptv 0.001 0.99000001 1))
      )
      (InheritanceLink (stv 0.99000001 0.99000001)
         (ConceptNode "friend@e780ee86-6a61-419c-a37e-a97d34c4f3eb" (stv 0.001 0.99000001))
         (ConceptNode "friend" (ptv 0.001 0.99000001 1))
      )
      (EvaluationLink (stv 0.99000001 0.99000001)
         (PredicateNode "mauled@1a06458b-e649-431c-9430-5940e43bbf21" (stv 0.001 0.99000001))
         (ListLink (stv 0.99000001 0.99000001)
            (ConceptNode "Bob@1c332525-f7a2-4b72-9256-9e32f0d5e9da" (stv 0.001 0.99000001))
            (ConceptNode "glockenspiel@b1288105-63bc-4630-bf0b-e35a585647ee" (stv 0.001 0.99000001))
            (ConceptNode "friend@e780ee86-6a61-419c-a37e-a97d34c4f3eb" (stv 0.001 0.99000001))
         )
      )
      (InheritanceLink (stv 0.99000001 0.99000001)
         (ConceptNode "with@8ef20cf6-fd2e-4f8f-ab0b-43129152e715" (stv 0.001 0.99000001))
         (ConceptNode "with" (ptv 0.001 0.99000001 1))
      )
      (InheritanceLink (stv 0.99000001 0.99000001)
         (SatisfyingSetLink (stv 0.99000001 0.99000001)
            (PredicateNode "mauled@1a06458b-e649-431c-9430-5940e43bbf21" (stv 0.001 0.99000001))
         )
         (ConceptNode "with@8ef20cf6-fd2e-4f8f-ab0b-43129152e715" (stv 0.001 0.99000001))
      )
      (InheritanceLink (stv 0.99000001 0.99000001)
         (PredicateNode "mauled@1a06458b-e649-431c-9430-5940e43bbf21" (stv 0.001 0.99000001))
         (ConceptNode "past" (stv 0.001 0.99000001))
      )
      (InheritanceLink (stv 0.001 0.99000001)
         (InterpretationNode "sentence@47794171-b923-41e5-a03a-7fc22eb583fb_parse_0_interpretation_$X")
         (ConceptNode "DeclarativeSpeechAct" (stv 0.001 0.99000001))
      )
      (EvaluationLink (stv 0.99000001 0.99000001)
         (PredicateNode "definite" (stv 0.001 0.99000001))
         (ListLink (stv 0.99000001 0.99000001)
            (ConceptNode "friend@e780ee86-6a61-419c-a37e-a97d34c4f3eb" (stv 0.001 0.99000001))
         )
      )
      (InheritanceLink (stv 0.99000001 0.99000001)
         (ConceptNode "his@3011b147-3e39-446d-8dee-b0b6fe67a2bf" (stv 0.001 0.99000001))
         (ConceptNode "his" (ptv 0.001 0.99000001 1))
      )
      (EvaluationLink (stv 0.99000001 0.99000001)
         (PredicateNode "possession" (stv 0.001 0.99000001))
         (ListLink (stv 0.99000001 0.99000001)
            (ConceptNode "friend@e780ee86-6a61-419c-a37e-a97d34c4f3eb" (stv 0.001 0.99000001))
            (ConceptNode "his@3011b147-3e39-446d-8dee-b0b6fe67a2bf" (stv 0.001 0.99000001))
         )
      )
      (ImplicationLink (stv 0.99000001 0.99000001)
         (PredicateNode "with@8ef20cf6-fd2e-4f8f-ab0b-43129152e715" (stv 0.001 0.99000001))
         (PredicateNode "with" (ptv 0.001 0.99000001 1))
      )
      (EvaluationLink (stv 0.99000001 0.99000001)
         (PredicateNode "with@8ef20cf6-fd2e-4f8f-ab0b-43129152e715" (stv 0.001 0.99000001))
         (ListLink (stv 0.99000001 0.99000001)
            (ConceptNode "friend@e780ee86-6a61-419c-a37e-a97d34c4f3eb" (stv 0.001 0.99000001))
         )
      )
      (InheritanceLink (stv 0.99000001 0.99000001)
```

```
      (SpecificEntityNode "Bob@1c332525-f7a2-4b72-9256-9e32f0d5e9da" (stv 0.001 0.99000001))
      (ConceptNode "male" (stv 0.001 0.99000001))
   )
   (InheritanceLink (stv 0.99000001 0.99000001)
      (SpecificEntityNode "Bob@1c332525-f7a2-4b72-9256-9e32f0d5e9da" (stv 0.001 0.99000001))
      (ConceptNode "Bob" (ptv 0.001 0.99000001 1))
   )
   (EvaluationLink (stv 0.99000001 0.99000001)
      (PredicateNode "definite" (stv 0.001 0.99000001))
      (ListLink (stv 0.99000001 0.99000001)
         (ConceptNode "Bob@1c332525-f7a2-4b72-9256-9e32f0d5e9da" (stv 0.001 0.99000001))
      )
   )
)
)
)
```

和表示”Jim bought a thimble at the store.”的超图

```
((ReferenceLink
   (InterpretationNode "sentence@d92a6c1d-bc69-4501-8ea9-e4430fc56296_parse_0_interpretation_$X")
   (SetLink
      (ImplicationLink (stv 0.99000001 0.99000001)
         (PredicateNode "bought@7e4b5f65-7a73-4776-94ac-156a5c3799e1" (stv 0.001 0.99000001))
         (PredicateNode "buy" (ptv 0.001 0.99000001 1))
      )
      (InheritanceLink (stv 0.99000001 0.99000001)
         (ConceptNode "Jill@bc320f63-6edf-49c4-ad1c-31ad932003f2" (stv 0.001 0.99000001))
         (ConceptNode "Jill" (ptv 0.001 0.99000001 1))
      )
      (InheritanceLink (stv 0.99000001 0.99000001)
         (ConceptNode "thimble@8a32736f-7344-44ad-b4a4-df34553954bc" (stv 0.001 0.99000001))
         (ConceptNode "thimble" (ptv 0.001 0.99000001 1))
      )
      (InheritanceLink (stv 0.99000001 0.99000001)
         (ConceptNode "store@ef655d95-4b5d-4df4-bd12-d6af443d4bec" (stv 0.001 0.99000001))
         (ConceptNode "store" (ptv 0.001 0.99000001 1))
      )
      (EvaluationLink (stv 0.99000001 0.99000001)
         (PredicateNode "bought@7e4b5f65-7a73-4776-94ac-156a5c3799e1" (stv 0.001 0.99000001))
         (ListLink (stv 0.99000001 0.99000001)
            (ConceptNode "Jill@bc320f63-6edf-49c4-ad1c-31ad932003f2" (stv 0.001 0.99000001))
            (ConceptNode "thimble@8a32736f-7344-44ad-b4a4-df34553954bc" (stv 0.001 0.99000001))
            (ConceptNode "store@ef655d95-4b5d-4df4-bd12-d6af443d4bec" (stv 0.001 0.99000001))
         )
      )
      (InheritanceLink (stv 0.99000001 0.99000001)
         (ConceptNode "at@50d120ad-3c57-4bed-a464-9c8542a787ab" (stv 0.001 0.99000001))
         (ConceptNode "at" (ptv 0.001 0.99000001 1))
      )
      (InheritanceLink (stv 0.99000001 0.99000001)
         (SatisfyingSetLink (stv 0.99000001 0.99000001)
            (PredicateNode "bought@7e4b5f65-7a73-4776-94ac-156a5c3799e1" (stv 0.001 0.99000001))
         )
         (ConceptNode "at@50d120ad-3c57-4bed-a464-9c8542a787ab" (stv 0.001 0.99000001))
      )
      (InheritanceLink (stv 0.99000001 0.99000001)
         (PredicateNode "bought@7e4b5f65-7a73-4776-94ac-156a5c3799e1" (stv 0.001 0.99000001))
         (ConceptNode "past" (stv 0.001 0.99000001))
      )
      (InheritanceLink (stv 0.001 0.99000001)
         (InterpretationNode "sentence@d92a6c1d-bc69-4501-8ea9-e4430fc56296_parse_0_interpretation_$X")
         (ConceptNode "DeclarativeSpeechAct" (stv 0.001 0.99000001))
      )
      (EvaluationLink (stv 0.99000001 0.99000001)
         (PredicateNode "definite" (stv 0.001 0.99000001))
         (ListLink (stv 0.99000001 0.99000001)
            (ConceptNode "store@ef655d95-4b5d-4df4-bd12-d6af443d4bec" (stv 0.001 0.99000001))
         )
      )
      (ImplicationLink (stv 0.99000001 0.99000001)
         (PredicateNode "at@50d120ad-3c57-4bed-a464-9c8542a787ab" (stv 0.001 0.99000001))
```

```
      (PredicateNode "at" (ptv 0.001 0.99000001 1))
    )
    (EvaluationLink (stv 0.99000001 0.99000001)
      (PredicateNode "at@50d120ad-3c57-4bed-a464-9c8542a787ab" (stv 0.001 0.99000001))
      (ListLink (stv 0.99000001 0.99000001)
        (ConceptNode "store@ef655d95-4b5d-4df4-bd12-d6af443d4bec" (stv 0.001 0.99000001))
      )
    )
    (InheritanceLink (stv 0.99000001 0.99000001)
      (SpecificEntityNode "Jill@bc320f63-6edf-49c4-ad1c-31ad932003f2" (stv 0.001 0.99000001))
      (ConceptNode "female" (stv 0.001 0.99000001))
    )
    (InheritanceLink (stv 0.99000001 0.99000001)
      (SpecificEntityNode "Jill@bc320f63-6edf-49c4-ad1c-31ad932003f2" (stv 0.001 0.99000001))
      (ConceptNode "Jill" (ptv 0.001 0.99000001 1))
    )
    (EvaluationLink (stv 0.99000001 0.99000001)
      (PredicateNode "definite" (stv 0.001 0.99000001))
      (ListLink (stv 0.99000001 0.99000001)
        (ConceptNode "Jill@bc320f63-6edf-49c4-ad1c-31ad932003f2" (stv 0.001 0.99000001))
      )
    )
  )
)
)
```

我们的下一步研究会将逻辑推理也加入成本计算的过程，从而进一步改进上述针对查询处理的超图匹配。假如知识库中代表下面这个句子的超图表示“Jim bought a musical instrument at the store.”(Jim 在商店买了一个乐器。)，那么如果该查询系统能做出以下推理：

```
InheritanceLink
ConceptNode "glockenspiel"
ConceptNode "musical instrument"
```

得到“钟琴”是一种“乐器”，那么会对“钟琴”改为“乐器”的修改操作赋值一个较低的成本值。基于这样的简单推理，系统会认为“Jim bought a musical instrument at the store.”（Jim 在商店买了一个乐器）比“Jim bought a power tool at the store.”（Jim 在商店买了一个电源工具）更符合上述查询。

8.2 基于后向链接推理的问答系统

上一节中介绍的基于超图匹配的问答虽然能回答不少的查询，但是随着知识库 Atomspace 的不断变大，搜索匹配的工作会变得越来越复杂，也越来越耗时。而且基于超图匹配的方法只是一个粗糙的启发式搜索过程，不适用于这些情况：对查询时间要求高、知识库数据不足或者在知识库中的答案需要经过一

定推理才能找到等。例如，该方法无法将查询“Who bought a glockenspiel?”（谁买了钟琴？）从下面的句子中找到近似匹配，但是这些句子都在一定程度上暗示了此查询的答案。

- Bob brought his new instrument home from the mall and played it for his kids.（Bob 从商场带了新乐器回家，并为他的孩子们演奏了几曲。）
- Ling-ling's house is like a museum of obscure musical instruments.（玲玲的家像一个稀有乐器博物馆）
- Jack always buys whatever he sees in a magazine. ... On the flight back from Guilin, there was nothing for Jack to read but a catalog of weird musical instruments.（Jack 总爱买任何他在杂志上看到的东西。在从桂林回来的飞机上，只给 Jack 提供了一本稀奇乐器相关的杂志）

从另一方面来说，经过在适当的知识库上推理，也不难从上面的句子中找出适合此查询的答案。例如，如果想从第二个句子“Ling-ling's house is like a museum of obscure musical instruments.”中找出符合查询的答案，我们可通过如下推理得到：

```
Ling-ling's house is like a museum of obscure musical instruments
(玲玲的家像一个稀有乐器博物馆。)

The glockenspiel is obscure, and the glockenspiel is a musical instrument
(钟琴很稀有，而且钟琴是一种乐器。)

A museum of entities of type X, contains many instances of X
(有关 X 的博物馆，含有很多 X 的实例。)

If X is in Y's house, then often Y has bought X
(如果 Y 的家中有 X，那么通常 Y 买了 X。)

|-

It is non-trivially probable that Ling-ling has bought a glockenspiel
(玲玲很有可能买了一个钟琴。)
```

在 Atomspace 中，这样的定性推理可以通过不同很多方式进行。下面我们将解释其中一种方式的推理过程。推理的前提可表示如下:

```
Ling-ling's house is like a museum of obscure musical instruments:
(玲玲的家像一个稀有乐器博物馆。)

InheritanceLink
ConceptNode "house_123"
ConceptNode "house"

EvaluationLink
PredicateNode "own"
ConceptNode "Ling-ling"
ConceptNode "house_123"

SimilarityLink
ConceptNode "house_123"
ConceptNode "museum_55"

InheritanceLink
ConceptNode "museum_55"
ConceptNode "museum"

EvaluationLink
PredicateNode "of"
ConceptNode "museum_55"
ConceptNode "instruments_12"

InheritanceLink
ConceptNode "instruments_12"
ConceptNode "obscure"

InheritanceLink
ConceptNode "instruments_12"
ConceptNode "musical"

InheritanceLink
```

```
ConceptNode "instrument_16"
ConceptNode "instrument"
```

```
The glockenspiel is obscure, and the glockenspiel is a musical instrument:
(钟琴很稀有，而且钟琴是一种乐器。)
```

```
InheritanceLink
ConceptNode "glockenspiel"
ConceptNode "obscure"

InheritanceLink
ConceptNode "glockenspiel"
ConceptNode "instrument_15"

InheritanceLink
ConceptNode "instrument_15"
ConceptNode "musical"

InheritanceLink
ConceptNode "instrument_15"
ConceptNode "obscure"

InheritanceLink
ConceptNode "instrument_15"
ConceptNode "instrument"
```

```
A museum of entities of type X, contains many instances of X:
(有关 X 的博物馆，含有很多 X 的实例。)
```

```
ImplicationLink  <.95,.99>
ANDLink
InheritanceLink
$X
ConceptNode "museum"
EvaluationLink
PredicateNode "of"
$X
$Z
```

```
EvaluationLink
PredicateNode "many"
SatisfyingSetLink
ANDLink
InheritanceLink
$A
$Z
EvaluationLink
PredicateNode "in"
$A
X
```

```
If X is in Y's house, then often Y has bought X:
(如果 Y 的家中有 X，那么通常 Y 买了 X。)
```

```
ImplicationLink <.8,.9>
ANDLink
EvaluationLink
PredicateNode "in"
ListLink
$X
$Z
InheritanceLink
$Z
ConceptNode "house"
EvaluationLink
PredicateNode "own"
ListLink
$Y
$Z
EvaluationLink
PredicateNode "buy"
ListLink
$Y
$X
```

根据上面的推理前提，PLN 可以推理得到下面的结论：

```
EvaluationLink <s,c>
PredicateNode "buy"
ConceptNode "Ling-ling"
ConceptNode "glockenspiel"
```

真值 $< s, c >$ 的大小不仅取决于推理过程中的使用到的推理规则的具体参数，还取决于节点本身的概率大小（“glockenspiel”节点的概率值用于表示钟琴在这世界上有多常见，这有助于估算钟琴是稀有乐器的概率）。因此，即使推理的前提给出很高的强度和置信度的真值如 $< 1, .95 >$，推断得出的结论的真值也可能很低如 $< .05, .6 >$。不过由于推理的不确定性，得到的这样的低真值结论也是合理的。上面例子给出的前提是简化过的，比如，没有给出这样的事实:“博物馆通常有稀有的东西”，可表示如下:

```
ImplicationLink <.3,.8>
ANDLink
InheritanceLink
$X
ConceptNode "museum"
EvaluationLink
PredicateNode "in"
$X
$Z
InheritanceLink
$Z
ConceptNode "obscure"
```

对该事实的考虑将在一定程度上增加结论的强度。总的来说，在推理链中不同阶段选择不同的前提，导致可以通过很多不同推理路径得到一个推理结论。每一条推理规则的执行是精确的，但对于选择哪条推理链则是一个非常模糊的问题，需要知识库 Atomspace 中的具体知识来指导。目前我们的逻辑系统 PLN 能完成上面的推理，但使用自然语言输入来进行这样的推理还存在一些问题。

8.3 一个综合的问答规划器

综合上述对问答规划的讨论，我们还设计并实现了一个合理的综合问答规划器，其实现方法如下：

Composite Information-Finding QA Schema:

给定一个用于信息查找的问题Q，回答Q的动机，以及所允许的最长等待时间T

如果T的值大，则试图通过逻辑推理来回答Q
    如果成功，则将得到的原子集合输入微观规划器用于应答
    如果失败，则通过基于超图的模糊匹配来回答Q
        如果成功，则将模糊匹配得到的原子集合来做为种子进行进一步推理，试图再次通过逻辑推理来回答Q
            如果成功，则将得到的原子集合输入微观规划器用于应答
            如果失败，则将模糊匹配得到的原子集合输入微观规划器用于应答
        如果失败，则触发StumpedByQuestion规划器

如果T的值小，则试图通过浅层推理来快速回答Q
    如果成功，则将得到的原子集合输入微观规划器用于应答
    如果失败，则试图通过基于超图的模糊匹配来回答Q
        如果成功，则将得到的原子集合输入微观规划器用于应答
        如果失败，则触发StumpedByQuestion规划器

其中，StumpedByQuestion Schema可规划如下：

StumpedByQuestion Schema:

给定一个问题Q（其中Q为智能对话系统试图回答但未成功）

当Q的重要值低时，
    触发ExpressIgnorance规划器

当Q的重要值很高时，
    使用PLN中的后向链推理找到P，使得

```
        P ==> Q
其中，P 相对简洁，且上述蕴含推理的真值很高

    如果找到合适的 P，将 P 标记为一个问题，
        并输入到微观规划器用于应答
    如果没有找到合适的 P，则
        触发 ExpressIgnorance 规划器
```

其中，ExpressIgnorance 规划器可规划如下：

```
ExpressIgnorance Schema:

将用于表示"I don't know"  或者
    "I don't know the answer to Q"的原子集合输入微观规划器用于应答
```

上面例子说明了 CogDial 方法的本质，我们可以将自适应的语篇管理能力分为下面三个阶段：

(a) 用 Scheme 或者 Python 编写相应语篇管理行为代码，然后绑定在 GroundedSchemaNodes 里。OpenPsi 可以在适当的时候调用这些语篇管理行为。

(b) 用基于超图的知识表示形式 Atom 来表示语篇管理行为，通过“硬编码”设置规划器参数，使其调用特定的认知处理程序（如微观规划器 Microplanner 或者后向链接推理工具 Backward Chainer 等）。

(c) 根据经验，通过强化学习和模仿等方法来自动学习语篇管理行为。

很显然，第三种方法是我们最终想要的能进行智能灵活对话的智能会话系统，但也无疑是最难的一种。我们 CogDial 首先实现第一种方法，然后在此基础上第二和第三种方法。这种增量的开发方式使我们能逐步实现智能的对话处理。使用硬编码的对话管理规划器，能使我们一种很直接的方式不断改进自然语言理解、生成和推理系统中与对话相关的处理，而不是一开始就将这些

可能不能处理复杂对话的自然语言处理和推理系统集成到复杂的自动学习对话管理中。当这些模块逐渐完善后，我们将会使用自动学习得到的对话管理规则来替换我们目前使用的手工编写的对话管理规划器。

对于是非（真值查询）问题的应答，可用下面的类似方法来处理：

```
Composite Truth-value QA Schema:

给定一个真值查询问题Q，回答该问题的动机，以及所允许的最长等待时间T

试图通过逻辑推理来回答Q
    如果成功，则将得到的原子集合输入微观规划器用于应答
    如果失败，试图通过基于超图的模糊匹配来回答Q
        如果成功，则将模糊匹配得到的原子集合来做为种子进行进一步推理，试图再次通过逻辑推理来回答Q
            如果成功，则将得到的原子集合输入微观规划器用于应答
            如果失败，则将模糊匹配得到的原子集合输入微观规划器用于应答
        如果失败，则触发StumpedByQuestion规划器


Composite Choice QA Schema:

给定一个选择查询问题Q，供选择的答案C1-Ck，回答该问题的动机，以及所允许的最长等待时机T

重复以下步骤直到T被用完，或者找到一个置信度高的有效答案
    对k个不同的查询“Ci是否为Q的答案”进行并行评估，并对第i个查询赋上其相关的资源r_i
    使用k臂赌博机算法持续重新计算资源分配r_i

如果上述循环返回一个置信度高的有效答案，则将得到的原子集合输入微观规划器用于应答
如果上述循环完成，但没有返回置信度高的有效答案，
    则触发StumpedByQuestion规划器
```

8.4　实例分析

该智能会话系统目前还处于非常初步的阶段，并没有完全实现我们前面章节列出的所有言语规划器和对话控制机制。目前还无法被用做一个很实用的对话系统，但我们认为，在上述的理论分析研究基础上，在完整实现不同的对话控制机制和言语规划器后，CogDial 的潜在推理和学习能力能使其成为一个有实用价值的智能会话系统。在本文撰写时，该系统还主要集中在问答机制的实现。本小节给出了一些目前已经实现的简单的对话结果。

- **CogDial:** Hi there
  - 所使用的言语规划器：开场白（Conventional-Opening）
- **Ruiting:** Hi
- **CogDial:** What do you want to talk about today?
  - 所使用的言语规划器：开放问题（Open-Question）
- **Ruiting:** What do YOU want to talk about today?
- **CogDial:** I’m thinking.
  - 所使用的言语规划器：自言自语（Self-talk）
- **Ruiting:** Do people really seek happiness?
- **CogDial:** Do you seek happiness?
  - 所使用的言语规划器：是非问答（Yes-No Question）
- **Ruiting:** Sometimes...
- **CogDial:** Hmmm...
  - 所使用的言语规划器：转移话题（Turn-Exit）
- **Ruiting:** Do you have any questions?
- **CogDial:** I’m tired of asking quesitons.
  - 所使用的言语规划器：是非问答（Yes-No Question）

8.5 深层语义解析的挑战

在研究和设计智能对话系统中，最具挑战性的并不是上述的复杂推理过程。在给定大知识库 Atomspace 的情况下，上述的推理过程会变得相当困难，但是我们可以通过在 OpenCog 中实现的注意力经济分配算法（EConomic Attention AllocatioN, 以下简称 ECAN）[?] 来对推理过程进行实时动态规划和控制，虽然 ECAN 目前仍需进一步的研究和改进，但从我们目前的实验数据来看，该方法是很可行的。在智能对话系统中，比复杂的推理更具有挑战性的问题是，如何使 Atomspace 获取上述问答规划器中的推理前提的超图表示形式。简单的说，这个挑战也就是如何将自然语言转换成合理准确的语义解析，这恐怕也是实现高度智能化的自然语言对话系统过程中最棘手的问题了。

例如，一个近乎准确的事实“如果 A 继承 Z，那么有很多 A 在 Z 的博物馆里”可表示如下：

```
ImplicationLink <.95,.99>
        ANDLink
                InheritanceLink
                        $X
                        ConceptNode "museum"
                EvaluationLink
                        PredicateNode "of"
                        $X
                        $Z
        EvaluationLink
                PredicateNode "many"
                SatisfyingSetLink
                        ANDLink
                                InheritanceLink
                                        $A
                                        $Z
                                EvaluationLink
                                        PredicateNode "in"
                                        $A
                                        $X
```

当前的自然语言系统不太可能获取上述例子中想要表达的推理形式。目前的句法分析或语义分析工具也无法根据字典中对博物馆（museum）的解释（如下）来得到上述的推理表达形式。

museum: a building in which objects of historical, scientific, artistic, or cultural interest are stored and exhibited.
(博物馆：珍藏和陈列历史、科学、艺术或者文化遗产的建筑)

但是上述的推理形式可以从自然语言处理系统可以理解的各种句子中推理得出，例如：

The New York Museum of Modern Art has one of the world's largest collections of French Impressionist works
(纽约的现代艺术博物馆是全球收藏法国印象派作品最多的博物馆之一。)

The British Museum boasts the world's largest Egyptology exhibit, with over five hundred sarcophagi on display,
along with some of the most perfectly preserved examples of engraved hieroglyphics.
(大英博物馆以全球珍藏最多古埃及文物闻名。其中有超过500个石棺，有些还带有保存最完整的象形雕刻文字。)

The US Postal Service Museum hosts enough rare stamps to satisfy even the most avid numismatist, but also
a variety of surprisingly interesting displays recounting, for example, the roles played by various animals in
the history of the postal service.
(美国邮政博物馆拥有足够多的稀有邮票，足以满足即便是最热心的徽章收藏家。不仅如此，该博物馆还展示很多出奇又有趣的历史回放，比如：各种动物扮演的在邮政历史上的不同角色。)

通过这些句子提供的信息，得到上述的推理形式就不是特别困难。只需要将这些“大（large）”“足够（enough）”“各种（variety）”等这些量词转换成一般概念层次上的“很多（many）”。而这样的关联完全可以从以下类似的句子中推理得出：

The New York Museum of Modern Art has one of the world's largest collections of French Impressionist works.
Many of these works are worth tens of millions of dollars, but their artistic value is incalculable.
(纽约的现代艺术博物馆是全球收藏法国印象派作品最多的博物馆之一。
其中很多作品都值上千万美金，但是他们的艺术价值是无价的。)

The dogs had more than enough steaks this morning. Many of them were rotten but they sure didn't seem to mind.
(狗狗们今天上午吃了太多牛排。其中很多牛排已经馊掉了，但是狗狗似乎一点也不介意。)

这些例句能很明确地将其他的量词和“很多（many）”联系起来。例如，解析上面最后一个例句后，RelEx2Logic 输出的语义逻辑关系中含有:

```
EvaluationLink
        PredicateNode "enough"
        ConceptNode "steaks_55"

EvaluationLink
        PredicateNode "many"
        ConceptNode "steaks_55"
```

在解析上面列出的第一个句子后，RelEx2Logic 输出的语义逻辑关系中含有:

```
 ImplicationLink
        PredicateNode "largest_77"
        PredicateNode "largest"

EvaluationLink
        PredicateNode "largest_77"
        ConceptNode "collection_44"

EvaluationLink
        PredicateNode "of"
        ConceptNode "collection_44"
        ConceptNode "works_88"
```

```
EvaluationLink
        PredicateNode "many"
        ConceptNode "works_88"
```

因此如果系统的知识库中含有以下推理常识：

```
ImplicationLink
        AndLink
                InheritanceLink
                        $C
                        ConceptNode "collection"
                EvaluationLink
                        PredicateNode "of"
                        ConceptNode "collection"
                        $W
                EvaluationLink
                        PredicateNode "many"
                        $W
        AndLink
                EvaluationLink
                        PredicateNode "many"
                        $C
```

那么就能推断出：

```
ImplicationLink
        PredicateNode "largest_77"
        PredicateNode "largest"

EvaluationLink
        PredicateNode "largest_77"
        ConceptNode "collection_123"

EvaluationLink
        PredicateNode "many"
        ConceptNode "collection_123"
```

如果系统中还有下列常识：

```
ImplicationLink
        PredicateNode "largest"
```

```
        PredicateNode "large"
```

那么就能推断出：

```
EvaluationLink
        PredicateNode "large"
        ConceptNode "collection_123"

EvaluationLink
        PredicateNode "many"
        ConceptNode "collection_123"
```

Putting these conclusions about enough vs. many and large vs. many, from these sentences, together with similar conclusions from other sentences, the (uncertain) conclusions 将这些从上面例句中推理得来的“大（large）”“足够（enough）”和“很多（many）”相关的结论联合起来，不难得出下列（非确定性）结论：

```
ImplicationLink
        EvaluationLink
                PredicateNode "large"
                ConceptNode $X
        EvaluationLink
                PredicateNode "many"
                ConceptNode $X

ImplicationLink
        EvaluationLink
                PredicateNode "enough"
                ConceptNode $X
        EvaluationLink
                PredicateNode "many"
                ConceptNode $X
```

这些可以看成是非确定性的启发式的蕴含规则，而这些不确定的直观的知识也正是很多常识性推理的基础。

上述推理中我们假设系统的知识库已存在一定的推理前提，如：

```
ImplicationLink
        AndLink
```

```
            InheritanceLink
                    $C
                    ConceptNode "collection"
            EvaluationLink
                    PredicateNode "of"
                    ConceptNode "collection"
                    $W
            EvaluationLink
                    PredicateNode "many"
                    $W
    AndLink
            EvaluationLink
                    PredicateNode "many"
                    $C
```

那么系统如何获取这些假定的直观常识呢，我们的猜想是可以通过自然语言理解系统解析一些相关句子得到，对于本例，系统可以通过解析下面句子获取相关常识:

```
His stamp collection is bigger than hers.
(他的邮票收藏库比她的大。)

The bigger bowl has many kinds of candy in it.
(大一点的那个碗里有很多种糖果。)
```

其中第一个句子中的“collection（收藏）”与“bigger（更大）”关联，而第二个句子中“bigger（更大）”与“many（很多）”关联，因此通过演绎推理可以得出“collection（收藏）”与“many（很多）”关联。

一般来说，解析某些特定句子所需要的知识都能从其他的句子中获得，但解析这些“其他的句子”可能需要从另外的其他的句子中获取相关知识。因此我们面临的挑战是如何构建一个强大的系统框架，使得获取推理常识的知识网络是一个良性而不是恶性循环。

这个挑战其中一部分是由组合爆炸引起，在我们的超图知识库 Atomspace 中包括很多不同句子的解析结果，这很容易导致过量的不同推断。上述例子中

的任何特定推断理论上都可以通过目标驱动的后向链接推理得出，如**??**中讨论的回答查询问题的例子“谁在商店买钟琴”中的推理模式，这样一来，我们需要设置 Atomspace 能储存相对久的知识以保证推理的时候有足够多的推理前提。同样，上述例子中的任何特地推断原则上也可以通过系统自动搜索相关有趣信息来进行前向链接推理得出。不论用哪一种推理方法，不仅需要一个非常丰富复杂的知识库的支持，如果合理高效地进行推理控制更是巨大的研究挑战。

这个挑战并不仅仅是针对查询处理，其实，更关键挑战基本上可以归结为常识推理的问题上[?]。考虑到即使是一个简单的常识推理，所涉及的知识领域和其复杂性不言而喻，纯粹通过手工编码构建的复杂知识库如 Cyc[?]已被证实并不可行。但是随着大数据处理技术的发展，通过分析和挖掘自然语言语料库以及智能体的涉身经验中的语言和非语言信息的综合数据库，来获取常识推理所需要的知识便成为一个更可行有效的方法。然而，根据自然语言的特性，某一个自然语言句子中用于推理的逻辑表现形式，常常与其他句子的表现形式不尽相同，因此通常需要一些从额外的句子中获取的常识来对这些句子进行逻辑形式重排，这也就要求一个可靠的能进行实时不确定逻辑推理系统。这无疑是一个相当棘手的问题。

要解决上述的常识知识问题，对于智能对话系统来说，其中一种方法是粗略模仿人类儿童的语言和认知成长轨迹。也就是说，从相对简单的知识库知识库着手，使得智能对话系统能通过知识库中相对简单的句子和人类交流相对简单的知识。当系统已经建立了基于这些相对简单的知识网络下的常识性知识后，再开始对系统输入稍微复杂的知识–等等，依此下去，这样的对话系统很可能会发展成一个拥有相当的常识语义库并能做出实时合理应答的智能对话系统。

从概念上讲，上述方法和[?]中的无监督语言学习类似，即通过逐步将越来越复杂的自然语言句子输入系统，使得系统逐步理解越来越复杂的句子。[?]也提出了类似的涉身行为学习。但我们这里提出的观点更针对智能对话系统，更注重如何从自然语言中抽取出有用的合理的语义知识等的学习。

8.6　本章小结

本章充分利用了我们的研究成果，在第4章中提出的智能会话系统 CogDial 的概念模型基础上，并结合第5章中讨论的自然语言理解系统、第6章中讨论的自然语言生成系统，以及2.3中讨论的 PLN 推理系统，以认知模型 OpenPsi 中的动机机制等技术来设计并实现了几个合理有效的对话管理机制。本章还讨论了我们在智能对话系统的研究中发现的深层语义解析的挑战，也给未来的研究提供了一个明确的方向。下一步工作将进一步完善本章中提出的各种对话管理机制并开发更多新颖智能的对话控制体系结构，并最终运用到机器人或者其他智能体软件系统上。

第九章　总结与展望

XXXX 这章需要重新写

9.1　研究工作总结

XXX 本文的主要贡献和创新，以及存在的问题。

9.2　下一步研究方向

XXX 下一步研究方向

我们最初是秉持创造出智能性自然语言对话系统之实务目标，着手进行本文研究的设计和开发。我们探讨的方法论是基于四个假设，如前言所述：

beginitemize

4. **假设 1——关于自然语言理解：**借助一个超图转换系统，使用依存关系语法、传统逻辑与谓词逻辑的合理结合，将各式各样的自然语言表达转换成满足下列要求的逻辑表达方式，是可行的。

    - 包含自然语言表达式中的主要语义。
    - 具体化自然语言表达式中存在的任何无法在语言到逻辑的转换消除的歧义，使得这些歧义能通过基于语境知识的逻辑推理后很直截了当地得到消除。

5. **假设 2——关于基于语言的推理：**借助由超图转换表示的推理规则和基于超图表示的知识库，使用简单的逻辑推理，在上述自然语言理解框架输出的逻辑表达式上实现基本的常识推理，是可行的。

6. **假设 3——关于自然语言生成：** 借助一个超图转换系统和一个超图匹配系统，在由自然语言理解系统自动生成的二元组（语言表达式，逻辑表达式）组成的知识库中根据逻辑表达式找到匹配的语言表达式并生成自然语言，是可行的。

7. **假设 4——关于智能会话系统：** 利用上述的语言理解、生成和逻辑推理系统，构建一个有用且灵活的智能会话系统，是可行的。

在本文研究的六年之中发现，为了创造真正可行的智能会话系统，首先必须对自然语言(NL) 的理解和生成投注大量心力。最终我们大部分的研发时间都耗费在创造足够广泛、强建稳固的自然语言理解系统，这后来作为我们自然语言生成系统的基础（通过经由反向理解达到完善化阶段语言生成）。然而，我们进行理解、产生和推理的试验，帮助我们透彻地思考设计对话系统的问题。我们在此文呈现的 CogDial 设计，远比初步的设计和现有的原型实现还要充实、缜密，虽然功能有限，但这让我们相信整体设计上是可行的。

本文的研究成果包含：

- 修复和改进链语法分析器和 RelEx 系统的诸多方面，并设计和实现了 RelEx2Logic 系统，使得自然语言能转换成逻辑形式。因此论证了假设 1 的正确性

- 论证演绎推理和针对比较级的推理；PLN 使用 RelEx2Logic 的输出来作为推理前提，能完成一定的常识推理，因此至少在单纯的案例中论证了假设 2 的正确性

- 设计并实行了微观规划器 Microplanner 和能将基于超图的逻辑表达转换成英文句子的表层生成器 SuReal，通过反向操作我们的自然语言理解通道(NL comprehension pipeline)，使得逻辑形式能转换成自然语言，从而架起了语言和逻辑之间的桥梁，也因此论证了假设 3 的正确性

- 根据言语行为理论和 OpenCog 框架，结合情感计算模型 OpenPsi 中的目标驱动的机制，提出了一个智能会话系统的具体设计；并使用设计的核心概念来

实行一个原型问答系统，进而证明此设计的可行性。这提供了假设 4 可能属实的初步迹象，但仍有许多工作需要完成。

- 提出了无监督语言学习的概念设计，为使用通过机器学习的规则来取代我们目前自然语言理解系统中使用的人工编写的规则库奠定了基础。

要实现创造出具有充分能力的智能会话系统之宏伟目标，仍有大量的研究工作待完成。但我们已在人工智能对话的每一个关键层面有实质上的进展，对于逻辑对话系统的每个元件如何运作、以及这些元件应如何一起运行，我们都学到了不少。我们也创造出有价值的软件工具，这些都正由开放源社区的其他人员积极的改良和发展，最终很有可能被充分发挥、利用在实际产品。在许多不同层面上，我们完成的研究为往后更多智能对话系统的研发打下了实用与概念的基础。

博士期间发表的论文

[1] 第一作者, 第二作者, 第三作者. 文章名一. 刊物, 2009, 08:10-13.

[2] 第一作者, 第二作者, 第三作者. 文章名二. 刊物, 2009, 09:20-23.

致　谢

谢谢父母

谢谢导师

谢谢朋友

谢谢同学

谢谢其他人