基于部分知识编译的近似模型计数方法：初步结果

报告人：赖　永

单　位：吉林大学

报告大纲

- 模型计数（#SAT）
- 知识编译
- Decision-DNNF
- 知识编译现存问题
- Partial Decision-DNNF
- 部分编译方法
- 初步实验结果

模型计数问题

- 定义：计算CNF公式的可满足的赋值数
- #SAT问题是#P-complete的，与贝叶斯推理中计算后验概率问题可在多项式时间内转化
- 2CNF公式也是#P-complete

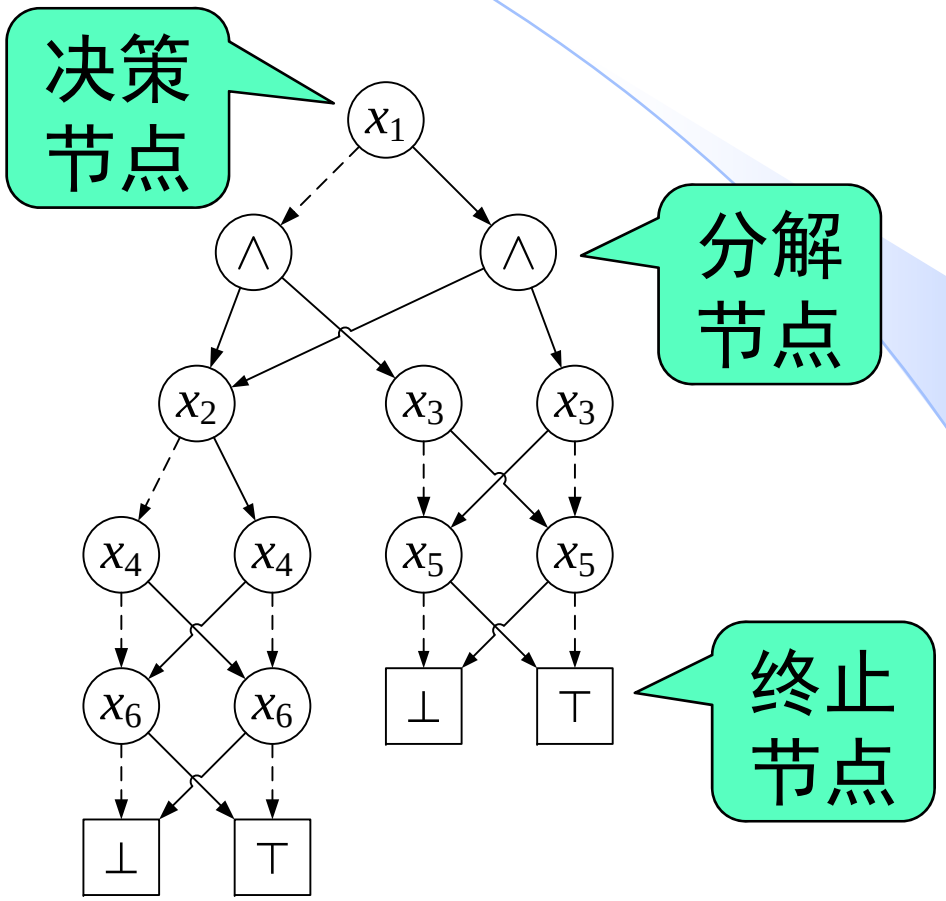
知识编译

- 离线阶段将知识库编译成易处理的形式，高效地支持在线阶段的推理任务
- 知识编译领域两大研究主题：
  - **目标语言及其评估标准**
  - **编译方法**

Decision-DNNF

- 目前绝大多数高效的编译器生成的都是Decision-DNNF或其子集
- 定义：有根的有向无环图，两类非终止节点，称为决策节点和分解节点，两类终止节点表示false和true。
- 决策节点有两个子节点，表示一个变量赋值为false和true后对应的子公式。
- 分解节点对应的子图不含公共变量。

Decision-DNNF例子



$$(x_1 \leftrightarrow x_3 \leftrightarrow x_5) \wedge (x_2 \leftrightarrow x_4 \leftrightarrow x_6)$$

Decision-DNNF模型计数

- 终止节点：0或2^n
- 如果u是分解节点且u的子节点$v_1, \ldots, v_m$,

$$\#models(u) = \frac{\prod_{i=1}^{m} \#models(v_i)}{2^{n(m-1)}}$$

- 如果u是决策节点且u的子节点v, w,

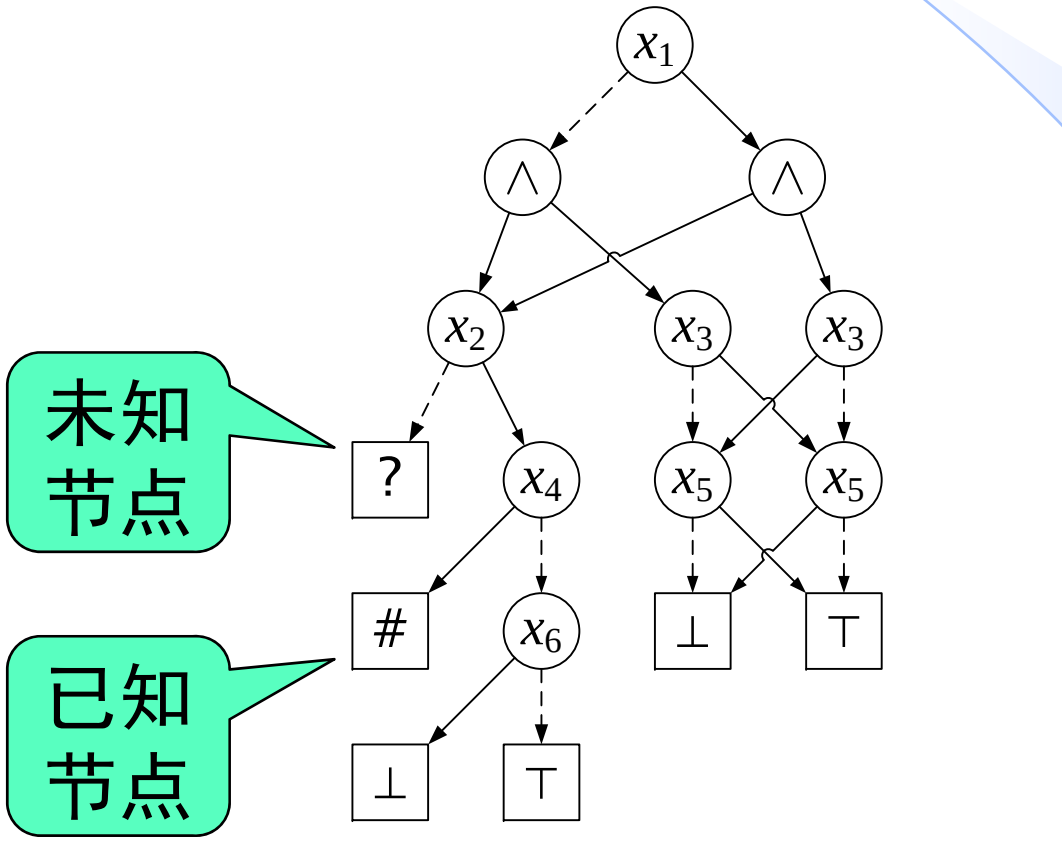
$$\#models(u) = \frac{\#models(v) + \#models(w)}{2}$$

完全知识编译的问题

- 完全编译算法对于低树宽的问题相对有效，对于高树宽的问题，通常无法完成编译，从而无法进行模型计数。
- 对于太复杂的问题，目前的一种做法是通过抽样获得近似的模型数，抽样的一个缺点是每次抽样之间独立，不能共用信息提高效率。

Partial Decision-DNNF

- 增加两类节点：已知节点⟨#⟩和未知节点⟨?⟩



$$(x_1 \leftrightarrow x_3 \leftrightarrow x_5) \wedge (x_2 \leftrightarrow x_4 \leftrightarrow x_6)$$

Partial Decision-DNNF模型计数

- 给定一个非未知节点u，已知子节点模型数的无偏估计，可以得到该节点模型数的无偏估计
  - **如果u是分解节点且u的子节点$v_1, \cdots, v_m$，**

$$\#models(u) = \frac{\prod_{i=1}^{m} \#models(v_i)}{2^{n(m-1)}}$$

  - **如果u是分解节点且u的子节点v, w，**

$$\#models(u) = \frac{\#models(v) + \#models(w)}{2}$$

部分知识编译方法

- 给定一个CNF公式对应的Partial Decision-DNNF *G*，每个节点对应一个部分赋值下的子公式
- 使用一个称为MicroCompile函数通过结合随机赋值和分解逐步编译CNF公式。
- 将MicroCompile得到的结果合成到*G*中
- 循环地调用MicroCompile逐步增大*G*

MicroCompile：三种情况

- 如果φ足够简单，直接调用已有的精确模型计数算法得到一个已知节点。

MicroCompile：三种情况

- 如果φ未出现在*G*中，选取一个变量随机赋值，得到φ的子公式ξ
- 计算ξ的Backbone，利用Backbone简化ξ，然后对简化后公式动态分解成若干个子公式，后递归地调用MicroCompile

MicroCompile：三种情况

- 如果φ已出现在*G*中，直接选取对应节点v，对节点标记的变量随机赋值。
- 如果v存在与赋值对应的非未知节点，直接获取Backbone和利用Backbone简化后的公式的动态分解情况。否则，计算ξ的Backbone，利用Backbone简化ξ，然后对简化后公式动态分解成若干个子公式。
- 对上一步得到的子公式递归地调用MicroCompile。

MicroCompile三个细节

- 通过判断CNF公式中变量数来判断公式模型计数的难易程度。
- 通过多次调用SAT求解器计算Backbone。
- 随机赋值时选择false和true的概率相同

初步实验结果

- 测试方法，利用马尔可夫不等式求下界

$$P(\frac{X}{\alpha} > M) < \frac{1}{\alpha}$$

instance	PartialKC		SampleSearch	
	#micro	result	#sample	result
ls13-norm	2	2.27E+44	6723	**1.15E+55**
lang16	51212	1.04E+08	14971	**6.51E+08**
9symml_gr_rcs_w6	31053	**1.55E+83**	6241	2.80E+82
apex7_gr_2pin_w5	1104	**4.13E+94**	48331	2.33E+94
c880_gr_rcs_w7	605	**3.25E+261**	831	7.16E+255
example2_gr_rcs_w6	3700	**2.80E+260**	6211	6.85E+250
vda_gr_rcs_w9	42	**2.55E+305**	221	5.08E+300

instance	#micro	time	avg-time
ls13-norm	2	7618	
lang16	10000	1099	
	20000	1544	**0.054**
	30000	1885	**0.034**
9symml_gr_rcs_w6	10000	1652	
	20000	2415	**0.076**
	30000	3022	**0.061**
apex7_gr_2pin_w5	300	1872	
	600	3653	5.94
	900	5736	6.94
c880_gr_rcs_w7	600	6830	
	1200	12187	8.93
	1800	16807	7.7
example2_gr_rcs_w6	1000	2843	
	2000	3991	1.148
	3000	5801	1.9

instance	PartialKC抽样次数	单独抽样次数
ls13-norm	2	2
lang16	51212	8230
9symml_gr_rcs_w6	31053	18531
apex7_gr_2pin_w5	1104	1067
c880_gr_rcs_w7	605	546
example2_gr_rcs_w6	3700	2541
vda_gr_rcs_w9	41	41

进一步改进的方向

- MicroCompile求Backbone调用多次SAT求解器，对于一些问题，求SAT时间过长，导致MicroCompile调用次数过少。
- 当子公式比较简单时，直接调用精确#SAT算法，怎么判断公式的难易程度是个难题。目前通过检测子公式中变量数目确定问题难易程度，为了程序的鲁棒性，精确#SAT算法只求解不多于256个变量的公式。
- 给定变量不同赋值后，得到不同子公式若模型数偏差过大，且模型数大的子公式出现概率很小，PartialKC收敛很慢。

谢谢大家