哈尔滨工程大学学报
Journal of Harbin Engineering University
ISSN 1006-7043,CN 23-1390/U

《哈尔滨工程大学学报》网络首发论文

题目：利用细胞膜演算描述带子句学习的 DPLL 算法
作者：李壮，刘磊，吕帅，任俊绮
收稿日期：2017-09-28
网络首发日期：2018-10-30
引用格式：李壮，刘磊，吕帅，任俊绮．利用细胞膜演算描述带子句学习的 DPLL 算法[J/OL]．哈尔滨工程大学学报．
http://kns.cnki.net/kcms/detail/23.1390.U.20181026.1442.006.html

利用细胞膜演算描述带子句学习的 DPLL 算法

李壮[1]，刘磊[1]，吕帅[1,2]，任俊绮[1]

（1. 吉林大学计算机科学与技术学院，长春 130012；2. 符号计算与知识工程教育部重点实验室（吉林大学），长春 130012）

摘　要：为了达到推理算法形式化描述的目的，采用细胞膜演算的形式化方法描述带子句学习的 DPLL 算法。分别定义了部分赋值、变元反转、回溯、回跳最大层、细胞膜溶解等反应规则，给出了 DPLL 的一般过程和冲突分析过程的描述。最后通过一个算例的求解过程验证了该形式化描述方法的可行性。依赖细胞膜演算可以更直观、简洁地展现推理算法的推理过程，同时展示了膜演算的描述能力和处理能力。

关键词：人工智能；问题求解；形式化方法；自动推理；DPLL；子句学习；演算；细胞膜演算

Doi：10.11990/jheu.201709122

中图分类号：TP181　　**文献标识码：**A

Using membrane calculus to describe DPLL algorithm with clause learning

LI Zhuang[1]，LIU Lei[1]，LÜ Shuai[1,2]，REN Junqi[1]

（1. College of Computer Science and Technology, Jilin University, Changchun, 130012, China；2. Key Laboratory of Symbolic Computation and Knowledge Engineering (Jilin University), Ministry of Education, Changchun 130012, China）

Abstract：In order to formally describe the reasoning algorithm, a formal method of membrane calculus is used to describe DPLL algorithm with clause learning. Some reaction rules are defined respectively, such as partial assignment, variable flip, backtracking, maximum backjumping level and membrane dissolution. The general process of DPLL and the process of conflict analysis are described. Finally, the feasibility of the formal method is verified by a benchmark solving process. Depending on the membrane calculus, the reasoning process of the DPLL algorithm can be displayed more intuitively and concisely, the descriptive abilities and processing abilities of the membrane calculus can be displayed at the same time.

Keywords: artificial intelligence; problem solving; formal method; automated reasoning; DPLL; clause learning; calculus; membrane calculus

1960 年，DPLL 算法作为求解 SAT 问题的完备算法被 Davis 和 Putnam 等人提出。先进的 SAT 求解器普遍基于回溯的 DPLL 算法，其中包含了子句学习（clause learning）等技术[1-3]。

在 DPLL 的改进过程中，子句学习发挥着重要的作用。将子句学习的概念引入到 DPLL 框架中，构成了冲突驱动的子句学习。随着子句学习技术的不断成熟，各种不同的学习机制也相继提出。利用子句学习技术的 Chaff、Glucose、COMiniSat[4-5]等求解器能够解决包含百万个变元和子句的工业算例，这是其它推理算法无法做到的[6]。

P 系统是一种细胞膜计算模型[7-8]。该模型相继出现了许多变型，其中多半表达能力都与图灵机等价。根据细胞新陈代谢的特点如：消融、分裂或创建等，可知膜系统的格局及它们之间转移的概念[9-10]。

细胞膜演算是基于细胞膜计算提出的形式化方法[11-12]。在膜计算的基础上，膜演算增加了移入、移出、传入和传出等细胞膜之间的反应规则。

本文利用细胞膜演算形式化描述带子句学习的 DPLL 算法的整个推理过程，分别给出了 DPLL 的一般过程和冲突分析过程的描述，为 DPLL 算法提供了一种新的形式化描述方法，同时该方法对于经典的 SAT 问题的研究产生了一定的推动作用。

1　伴随子句学习的 DPLL 算法

1.1 DPLL 过程与子句学习

如今，先进的求解器皆基于 DPLL 算法。经典的 DPLL 算法过程如下：对于合取范式 F 和部分赋值 α，若 α 满足 F，则该合取范式判定为 SAT；若 α 导致其中某个子句为空，则该合取范式判定为

收稿日期：2017-09-28
基金项目：国家自然科学基金项目（61300049，61502197，61503044，61763003），吉林省科技发展计划资助项目（20180101053JC）.
作者简介：李壮（1988−），男，博士生；
刘磊（1960−），男，教授、博士生导师。
通信作者：吕帅，E-mail：lus@jlu.edu.cn.

UNSAT。一个变元 x 通过启发式被α选中，α通过 x:=0 或 x:=1，会递归调用该判定程序两次。若某一次递归调用判定为 SAT，则该合取范式可满足；若两次递归调用都判定为 UNSAT，则该合取范式不可满足。

若在单元传播过程中发生了一次冲突，且α的子集α'是导致这个冲突的赋值，则将其保存到一个新的子句 C 并将其添加到公式中，在之后的搜索中若再次出现部分赋值α'时，可以更早地回溯。

Algorithm 1: DPLL with clause learning [13]
Input: CNF formula F
Output: A solution of F or UNSAT if F is not satisfiable

```
1   P←Unit propagate
2   If P assigns a value to every variable
3     return success
4   Else
5     P contains a conflict
6     choose a conflict graph G to find c under P and add it to F
7     rollback variable or back jump
```

上述算法为带子句学习的 DPLL 算法的一般过程。其中，第 6 行的子句学习过程即在真值赋值发生冲突时，分析冲突产生的原因，找到导致冲突的赋值序列并添加到子句集中，避免再次发生相同的冲突。

1.2 冲突分析

在求解过程中，变元的赋值序列可以通过蕴含图表示。图中每个顶点代表一个文字，正文字代表变元被赋值为 1，负文字代表变元被赋值为 0；边表示导致一个变元被赋值的原因。每一个决策变元对应着一个决策层[14]。

原始的 DPLL 算法具备较简单的冲突分析机制：求解器为每个决策变元存储一个标记，用作记录变元赋值次数。当出现空子句后，算法会检测当前标记：若标记一次，则强制反转当前赋值；若标记两次，则回溯到上一层并强制反转赋值。

非时序性回溯是基于蕴含图的更先进的冲突分析机制。通过对冲突的分析，学习得到一些子句并将其添加到子句集中，这个过程称为冲突驱动子句学习，用于在将来搜索过程中避免相同的错误发生。若返回的最大层为 0，意味着算法已经无法再回溯，即整个问题不可满足。

1.3 重启机制

当求解器在搜索过程中发生了 k（k>1）次冲突后，激活重启功能，求解器终止当前求解过程并返回到第 0 层，即重新求解该子句集。冲突上界由参数 k 来决定[15]。

2 细胞膜演算

2.1 基本概念

细胞膜演算主要包括：膜结构、对象和反应规则。膜结构表达多个事务的嵌套关系，对象有不同类，对象之间可用反应规则来描述[7-10]。

细胞膜演算模型定义如下：

定义 **1** 细胞膜是一个元组 $MM=(\sum,\mu,O_1,...,O_m,R_1,...,R_m,C)$，其中：

1）Σ 是一个有限非空的类型集合，包含所有出现对象的类型；

2）μ 是一个细胞膜结构，包含 m 个细胞膜；

3）$O_i(1\le i\le m)$是第 i 个细胞膜的对象多集；

4）$R_i(1\le i\le m)$是第 i 个细胞膜的反应规则多集；

5）C 是类型映射函数，定义为$o\in O_i\to\sum$。该函数定义对象 o 的类型 $C(o)\to\sum$。直观上讲，所有对象都是类型化的。

2.2 语法

定义类型集合为 Σ，其元素为 T，对象多集中的元素为 $a,b,c\ldots$，反应规则为 R，标识符为 $i,j,k\ldots$。细胞膜演算定义如下：

$$M,M'::=0\,|\,[_i O,R,M]\,|\,M,M'$$

$$O,O'::=0\,|\,a:T\,|\,OO'$$

$$R,R'::=0\,|\,O\to O'\,|\,O\to O'_{out}\,|\,O\to O'_{in(j)}\,|\,O\to\lambda\,|\,R\,|\,R'$$

$$\lambda::=\delta\,|\,\sigma k\,|\,\eta k\,|\,M$$

在该语法表述中，首先定义了一系列的细胞膜 M、M'等，然后定义细胞膜中的对象类 O、O'和反应规则类 R、R'。细胞膜用$[0[1][2]...]$的嵌套形式表达彼此间的关系，详细内容请参考文献[9]。

2.3 反应规则

下面为反应规则的作用和效果的定义（O,O',O''为对象多集，M,M',M'',M'''为细胞膜多集，R,R',R''为规则多集，i,j,k 为标识符）：

对象迁移规则：$O\to O'$。将对象 O 变为 O'。

$$[_i OO'',O\to O'\,|\,R',M]\to[_i O'O'',O\to O'\,|\,R',M]$$

对象传出规则：$O\to O'_{out}$。将对象传送到其父细胞膜。

$$[_i O',R,[_j O'O'',O\to O''_{out}\,|\,R',M']M]\to[_i O'O''',R,[_j O'',O\to O''_{out}\,|\,R',M']M]$$

对象传入规则：$O\to O'_{in(j)}$。将对象传送到其子细胞膜。

$$[_i OO',O\to O''_{in(j)}\,|\,R,[_j O''',R',M']M]\to[_i O',O\to O''_{in(j)}\,|\,R,[_j O''O''',R',M']M]$$

对象溶解规则：$O\to\delta$。将该细胞膜溶解，将其对象保留于父细胞膜中。

$$[_i O'',R,[_j OO',O\to\delta\,|\,R',M']M]\to[_i O'',O',R,M]$$

细胞膜移出规则：$O \to \sigma i$。将子细胞膜 i 从父细胞膜中移出，成为兄弟细胞膜。

$$\left[{}_i O'', R, \left[{}_j OO', O \to \sigma i \mid R', M'\right] M\right]$$
$$\to \left[{}_i O'', R, M\right]\left[{}_j OO', O \to \sigma i \mid R', M'\right]$$

细胞膜移入规则：$O \to \eta i$。将兄弟细胞膜 i 移入到其子细胞膜，成为子细胞膜。

$$\left[{}_i O'', R, M\right]\left[{}_j OO', O \to \eta i \mid R', M'\right]$$
$$\to \left[{}_i O'', R, \left[{}_j OO', O \to \eta i \mid R', M'\right] M\right]$$

细胞膜生成规则：$O \to M$。将创建一个细胞膜 M。

$$\left[{}_i OO', O \to M \mid R, M'\right]$$
$$\to \left[{}_i O', O \to M \mid R, MM'\right]$$

反应规则是区别细胞膜演算和细胞膜计算的根本要素。只有定义对象的反应规则，才能真正实现形式化描述的过程。

3 细胞膜演算描述子句学习

3.1 细胞膜演算规则

带有子句学习的 DPLL 算法包含部分赋值、变元反转、回溯、回跳等核心步骤。多个对象集合表示为 $l, l_i, \neg l_i, l_{i-1}, \neg l_{i-1}, l_t, \neg l_t, L$，多个细胞膜集合表示为 M、M'，多个反应规则集合表示为 R、R'，标识符表示为 i、j、k，溶解表示为 δ。

1）部分赋值（细胞膜的产生）

决策文字表示为 l_i，变元集表示为 L，子膜的反应规则表示为 R'，父细胞膜表示为 M，产生的新细胞膜表示为 M'，是 M 的子膜。即：在决策变元被赋值后，对象 l_i 触发了反应规则，并产生了新的子细胞膜 M'，其它不变。

$$\left[{}_i l_i L, l_i \to M' \mid R', M\right]$$
$$\to \left[{}_i L, l_i \to M' \mid R', MM'\right]$$

2）变元反转

变元反转 l_i 后的值表示为$\neg l_i$。即：在发生冲突后，决策变元的赋值被强制反转。

$$\left[{}_i l_i L, l_i \to \neg l_i \mid R, M\right]$$
$$\to \left[{}_i \neg l_i L, l_i \to \neg l_i \mid R, M\right]$$

3）回溯

上一层的赋值被强制反转后表示为$\neg l_{i\text{-}1}$。同层冲突连续产生两次，需要返回到上一决策层并使该层的决策变元赋值强制取反。

$$\left[{}_i l_i L, l_i \to \neg l_{i-1} \mid R, M\right]$$
$$\to \left[{}_i \neg l_{i-1} L, l_i \to \neg l_{i-1} \mid R, M\right]$$

4）回跳到最大层

回跳最大层的决策变元表示为 l_t，强制反转最大层变元表示为$\neg l_t$。出现空子句后，原始的 DPLL 会回溯到上一层。带子句学习的 DPLL 包含冲突分析的过程，当学习子句被添加到子句集后，同时回跳到决策最大层并强制反转变元的赋值。

$$\left[{}_i l_i L, l_i \to \neg l_t \mid R, M\right]$$
$$\to \left[{}_i \neg l_t L, l_i \to \neg l_t \mid R, M\right]$$

5）细胞膜溶解

由部分赋值规则可知，决策变元赋值后会产生新的细胞膜。赋值反转后同样会产生新的细胞膜，而原有的细胞膜会被溶解。

$$\left[{}_i L, R, \left[{}_j l, l \to \delta \mid R' M'\right] M\right]$$
$$\to \left[{}_i L, R, M\right]$$

3.2 细胞膜演算描述 DPLL

下面来描述 DPLL 的一般过程。如图 1，首先定义对象集，决策类表示为 p，单元传播类表示为 q，反转类表示为 r，冲突类表示为 v，子句集表示为 S。最外层的细胞膜表示为 M_0，主要用于判定该层的冲突次数。

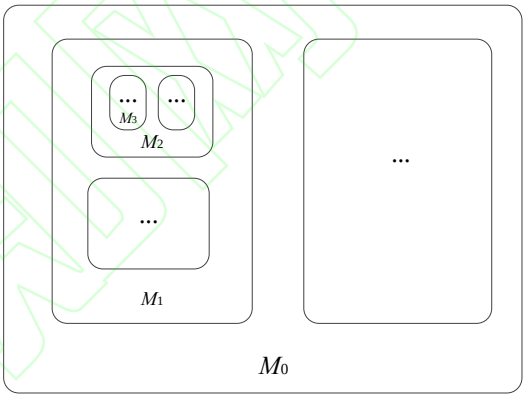

图 1 DPLL 过程对应的细胞膜嵌套关系

Figure 1 Nested relationship of membranes corresponding to DPLL process

$$O = (p_i, \neg q_i, \neg q_{i-1}, q, r, v, \neg v, \varnothing, S, T_p, F_p, T_q, F_q, y)$$
$$M_0 = \left[S, R_1, M_j\right]$$
$$R_0 = \left(r_{01}, r_{02}, r_{03}, r_{04}, r_{05}\right)$$
$$r_{01} : p_i \to T_p \mid F_p; T_p \to T_p M_{11} \mid F_p \to F_p M_{10}$$
$$r_{02} : q \to T_q \mid F_q$$
$$r_{03} : T_p \to T_{pinM11}; F_q \to F_{pinM10}$$
$$r_{04} : (v, \neg v) \to (v, \neg v)_{in(j)}, y \to \delta$$
$$r_{05} : S \to \varnothing_{out}$$
$$\rho_0 = \left\{r_{04} > r_{02} > r_{01} > r_{03} > r_{05}\right\}$$
$$R_j = \left\{r_{j1}, r_{j2}\right\}$$
$$r_{j1} : p_i \to \neg p_{iout}$$
$$r_{j2} : \neg p_i \to \neg p_{i-1out}$$
$$\rho_j = r_{j1} > r_{j2}$$

DPLL 求解首先对决策变元进行赋值。$r_{01} : p_i \to T_p \mid F_p; T_p \to T_p M_{11} \mid F_p \to F_p M_{10}$ 对决策变元 p 进行赋值，若决策变元 p 赋值为 T，则会产生新的子细胞膜 M_{11}。同理，若决策变元 p 赋值为 F，会产生另一个新的子细胞膜 M_{10}。

决策变元赋值后子句集可能会产生部分单元子句，这时进行单元传播，即所有单文字都用 q 来代替。在 $r_{02}:q\to T_q\,|\,F_q$ 中，q 进行决策 T_q 或 F_q。

规则 $r_{03}:T_p\to T_{pinM11};F_q\to F_{pinM10}$ 将决策变元 T_p 传入新的子细胞膜 M_{11} 中，并把决策变元 F_p 传入子细胞膜 M_{10} 中。

在 $r_{04}:(v,\neg v)\to(v,\neg v)_{in(j)},y\to\delta$ 中，y 为冲突层细胞膜。当出现冲突 $(v,\neg v)$ 时，子句集剩余部分移入判定膜 $(v,\neg v)_{in(j)}$，细胞膜 $y\to\delta$ 溶解。

最后一条 $r_{05}:S\to\varnothing_{out}$ 表示子句集可满足。

规则的优先级顺序同样存在于父细胞膜 M_0 中。其顺序用 $\rho_0=\{r_{04}>r_{02}>r_{01}>r_{03}>r_{05}\}$ 表示。反应规则必须按照优先级的排列顺序执行，当不符合当前规则的触发条件时，会跳转执行下一条规则的判定。

细胞膜 M_j 的判定仅有两条规则。该层的冲突次数在该层的子细胞膜中判定。在规则 $r_{j1}:p_i\to\neg p_{iout}$ 里，i 是 p_i 中冲突层的层数。当决策层发生冲突时，触发规则 r_4，对象传入判定膜。这里我们默认这是第一次冲突，因此只需强制反转 p_i，然后传入到原细胞膜并继续上述操作。

发生第二次冲突时触发规则 $r_{j2}:\neg p_i\to\neg p_{i-1out}$。由于第一次冲突产生的 p_i 已经被强制取反为$\neg p_i$，所以当该层再次发生冲突时，返回决策层上一层并将决策变元的赋值改写为$\neg p_{i\text{-}1}$，所以该规则在此时被触发。

在有大量子句的子句集中，随着赋值变元增多，二叉树搜索也会相应变深。所以在这里，我们可以用 M_{ij} 代替决策变元。

3.3 冲突分析

冲突分析是找出引起冲突的变元并更改其赋值。回跳也被称为非时序性回溯，是基于冲突驱动子句学习所提出来的方法。经过分析后得知准确的回跳层数，并将学习到的子句添加到子句集中。

子句学习属于典型的长事务实例。这个过程中，当开始执行单元传播时，如果产出了空子句，即触发回跳规则，找出最大层并返回到相应层。根据细胞膜演算，我们得到了相应的形式化描述。

$$MC=[Begin:State,R1,M0,ML,MJ]$$

$$M0=[{}_0\,0,R0,0]$$

$$ML=[{}_L\,0,RL,0]$$

$$MF=[{}_F\,0,RF,0]$$

$$MJ=[{}_J\,0,RJ,0]$$

$$R_1=$$

$$Begin:State\to Begin0:State_{in(0)}\,|$$

$$Success0:State\to Begin1:State_{in(L)}\,|$$

$$SuccessL:State\to BeginF:State_{in(F)}\,|$$

$$Fail0:State\to Learning0:State_{in(L)}\,|$$

$$FailI:State\to Judge:State_{in(J)}\,|$$

$$Fail0':State\to Learning0':State_{in(F)}\,|$$

$$FailL:State\to Learning0:State_{in(F)}$$

$$R_0=$$

$$Begin0:State\to L_0:State\,|$$

$$L_0:State\to Success0:State_{out}\,|$$

$$L_0:State\to Fail:State_{out}$$

$$R_L=$$

$$BeginL_l:State\to L_l:State\,|$$

$$L_l:State\to SuccessL:State_{out}\,|$$

$$L_l:State\to Fail0:State_{out}\,|$$

$$Fail0:State\to Learning0:State_{out}\,|$$

$$Learning0:State\to Fail0:State_{out}$$

$$R_F=$$

$$BeginL_l:State\to L_F:State\,|$$

$$L_F:State\to Success:State_{out}\,|$$

$$L_l:State\to FailI:State_{out}\,|$$

$$Learning0':State\to Fail:State_{out}\,|$$

$$LearningL:State\to FailL:State_{out}$$

$$R_J=$$

$$BeginJ:State\to Fail_L:State\,|$$

$$FailI:State\to Fail0':State_{out}\,|$$

$$FailI:State\to FailL:State_{out}$$

冲突分析的子事务表示为细胞膜 *MC*，赋值第 0 层的子事务表示为细胞膜 *M*0，决策层的子事务表示为细胞膜 *ML*，冲突层的子事务表示为细胞膜 *MF*，判定层的子事务表示为细胞膜 *MJ*。在单元传播过程中，传播的成功和失败状态分别表示为 *Success*、*Fail*，学习的子句表示为 *Learning*，对冲突的判定表示为 *Judge*。

开始时，细胞膜 ζ 从对象 *Judge* 开始，将子句集定义为重写规则 $Begin:State\to Begin0:State_{in(0)}$，对象 *Begin* 被重写为 *Begin*0 并传入到子细胞膜 *M*0 中，即通过重写规则使子事务开始运行。若运行结果是成功的，则改写状态 *Success*0 并传入到子事务——细胞膜 *ML* 中；运行结果是失败的，则 *Fail* 并结束整个事务。若细胞膜 *ML* 成功，则传出到细胞膜 *MC*。

同上，若细胞膜 *MC* 是成功的，那么子句集传出到细胞膜 ζ 判定整个事务成功。若冲突发生在子细胞膜 *ML* 中，则规则 $L_l:State\to Fail0:State_{out}$ 会将 *Fail*0 传出到规则 $Fail0:State\to Learning0:State_{in(L)}$，随后将再次传入细胞膜 *ML* 中，当规则 *Learning*0 返回到第 0 层而无法再次回溯时，返回到细胞膜 ζ 并结束整个过程。若细胞膜 *MC* 中发生冲突时，则冲突 *FailI* 会传入到另一个子细胞膜 *J* 中。这个冲突 *FailI* 为蕴含

冲突，与 *Fail*0 冲突有着本质的不同：冲突 *FailI* 所包含的冲突，可能是决策层引起的冲突，亦可能是第 0 层引起的冲突。所以需要子细胞膜 *MJ* 对 *FailI* 进行判定。根据判定结果，若 *FailI* 是由某一决策层导致的，则将其状态判定为 *FailL*，并根据回溯机制传出到细胞膜 *ML*；若 *FailI* 是由第 0 层导致的，则根据回跳机制传出到细胞膜 *M*0。

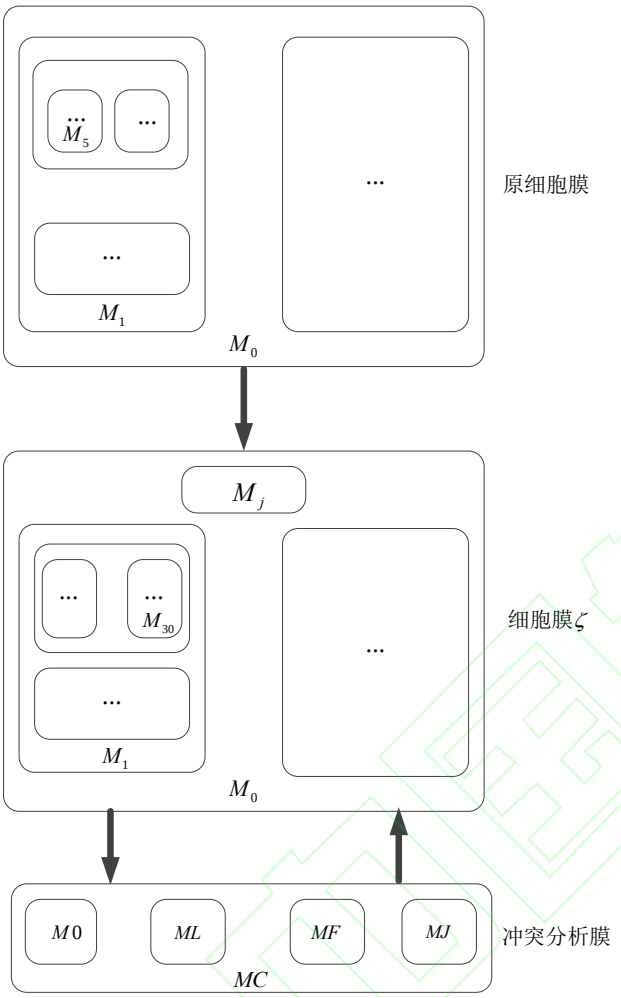

图 2 带子句学习的搜索过程

Figure 2 Searching process with clause learning

如图 2，假设冲突层是第 5 层，回跳层为第 3 层。带有子句学习的 DPLL 对决策变元赋值时，流程如下：当赋值存在于第 0 层时，当前层单元传播达到饱和状态后触发第 0 层产生第 1 层子细胞膜。当赋值到第 0 层时，以同样的顺序再传入第 2 层、第 3 层。当冲突出现在第 5 层时，冲突信息将由当前层反馈到上一层，随后强制反转当前层的决策赋值并再次进行单元传播。若第二次出现冲突，结束该膜的所有操作并传入冲突分析膜，进行冲突分析。

4 算例验证

本节对于例 1 所示的 SAT 算例展示经典的求解过程和细胞膜演算的形式化描述。

例 1. $\begin{bmatrix} p \vee q \vee r \\ s \vee \neg r \vee p \\ \neg s \vee \neg t \\ u \vee \neg q \\ t \end{bmatrix}$

该子句集包括 6 个变元、5 条子句，经求解后存在若干个解。若变元 *u* 和 *p* 同时取 *F*，便会导致一个冲突，从而使子句变得不可满足。

推理过程为：首先单元子句 *t*=*T*，通过单元传播，使得 *s*=*F*，进而子句集化简为：$\begin{bmatrix} p \vee q \vee r \\ \neg r \vee p \\ u \vee \neg q \end{bmatrix}$。

随机选择变元 *u* 进行赋值 *u*=*F*。通过单元传播使得 *q*=*F*，进而子句集化简为：$\begin{bmatrix} p \vee r \\ \neg r \vee p \end{bmatrix}$。

若变元 *p* 赋值为 *p*=*F*，通过单元传播使得 *r*=*T*，所以¬*r*=*F*。此时因出现了一对互补文字而产生了空子句，所以子句集不可满足。然后，通过触发冲突分析机制，反转文字 *p* 的取值，子句集最终可以找到可满足解。

细胞膜描述过程为：

$$\begin{aligned}
&\left[{}_0t \mid S_0, t \to T, S_0 \to S_1, M\right] \\
&\to \left[{}_1T \mid S_1, t \to T, S_0 \to S_1, M\right] \\
&\left[{}_1s \mid S_1, s \to F, S_1 \to S_1', M\right] \\
&\to \left[{}_1F \mid S_1', S \to F, S_1 \to S_1', M\right] \\
&\left[{}_1u \mid S_1', u \to F, S_1' \to S_2, M\right] \\
&\to \left[{}_2F \mid S_2, u \to F, S_1' \to S_2, M\right] \\
&\left[{}_2q \mid S_2, q \to F, S_2 \to S_2', M\right] \\
&\to \left[{}_2F \mid S_2', q \to F, S_2 \to S_2', M\right] \\
&\left[{}_2p \mid S_2', p \to F, S_2' \to S_3, M\right] \\
&\to \left[{}_3F \mid S_3, p \to F, S_2' \to S_3, M\right] \\
&\left[{}_3r \mid S_3, r \to T, S_3 \to S_3', M\right] \\
&\to \left[{}_3T \mid S_3', r \to T, S_3 \to S_3', M\right] \\
&\left[{}_3(r, \neg r) \mid S_3', (r, \neg r) \to \varnothing, S_3 \to \varnothing, M\right] \\
&\to \left[{}_3\varnothing \mid \varnothing, S_3'(r, \neg r) \to \varnothing, S_3 \to \varnothing, M\right] \\
&\left[{}_2p \mid S_2', p \to T, S_2' \to S_3, M\right] \\
&\to \left[{}_2T \mid S_3', p \to F, S_2' \to S_3, M\right]
\end{aligned}$$

演化过程开始时，子句集产生于 M_0。当对象 *t* 赋值为真时，子细胞膜 M_1 产生并与对象 *t* 建立了对应关系。由于 *s* 是单元传播决定的变元，所以 *s* 是同层变元，此时并不产生新的细胞膜，即仍然在细胞膜 M_1 中。同上，当 *u* 赋值为假时，产生细胞膜 M_2。在 DPLL 过程中出现了互补文字对 $(r, \neg r)$，即发生了冲突。该冲突导致当前层决策变元 *p* 取反，原变元所属的细胞膜溶解。此时，通过单元传播使子句集为空，子句集可满足。

5　结论

1)采用细胞膜演算的形式化方法描述带子句学习的 DPLL 算法，分别定义了部分赋值、变元反转、回溯、回跳最大层、细胞膜溶解等反应规则，给出了 DPLL 的一般过程和冲突分析过程的描述。最后通过一个算例来验证该形式化方法是可行的。

2）选用计算机领域中最成功的 SAT 求解方法进行刻画，使得整个推理过程可以与细胞膜演算过程完全对应，展现了细胞膜演算的描述能力。

参考文献：

[1] Paul Beame, Henry A. Kautz, Ashish Sabharwal. Towards understanding and harnessing the potential of clause learning [J]. Journal of Artificial Intelligence Research, 2004, 22: 319-351.

[2] Knot Pipatsrisawat, Adnan Darwiche. On the power of clause-learning SAT solvers as resolution engines [J]. Artificial Intelligence, 2011, 175(2): 512-525.

[3] Gilles Audemard, Jean-Marie Lagniez, Bertrand Mazure, Lakhdar Saïs. On freezing and reactivating learnt clauses [C]. Proceedings of the 14th International Conference on Theory and Applications of Satisfiability Testing (SAT 2011), Ann Arbor, MI, USA, June 19-22, 2011, 188-200.

[4] Albert Atserias, Johannes Klaus Fichte, Marc Thurley. Clause-learning algorithms with many restarts and bounded-width resolution [J]. Journal of Artificial Intelligence Research, 2011, 40: 353-373.

[5] Said Jabbour, Jerry Lonlac, Lakhdar Sais, Yakoub Salhi. Revisiting the learned clauses database reduction strategies [DB/OL]. CoRR abs/1402.1956, 2014.

[6] Marijn Heule, Matti Järvisalo, Florian Lonsing, Martina Seidl, Armin Biere. Clause elimination for SAT and QSAT [J]. Journal of Artificial Intelligence Research, 2015, 53: 127-168.

[7] Qi Zhengwei, You Jinyuan. The formal specification and verification of transaction processing in web services by membrane calculus [J]. Chinese Journal of Computers, 2006, 29(7): 1137-1144.

[8] Qi Zhengwei, Li Minglu, Fu Cheng, Shi Dongyu, You Jinyuan. Membrane calculus: A formal method for Grid transactions [J]. Concurrency and Computation: Practice and Experience, 2006, 18(14): 1799-1809.

[9] 戚正伟, 尤晋元. 基于细胞膜演算的 Web 服务事务处理形式化描述与验证[J]. 计算机学报, 2006, 29(7): 1137-1144.
Qi Zhengwei, You Jinyuan. The formal specification and verification of transaction processing in web services by membrane calculus[J]. Chinese Journal of Computers, 2006, 29(7): 1137-1144.

[10] 戚正伟. 细胞膜演算: 一种新的事务处理形式化方法研究 [D]. 上海交通大学, 2005.
Qi Zhengwei. Membrane calculus：A new formal method for transaction processing [D]. Shanghai Jiao Tong University, 2005.

[11] 任俊绮, 刘磊, 张鹏. 适用于演化过程建模的通信膜演算 [J]. 哈尔滨工程大学学报, 2018, 39 (4) :751-759.
Ren Junqi, Liu Lei, Zhang Peng. Communication membrane calculus: A formal method for modeling the process of evolution [J]. Journal of Harbin Engineering University, 2018, 39 (4): 751-759.

[12] 刘磊, 刘丰, 吕帅, 任俊绮. 基于细胞膜演算的 Dryad 形式化描述 [J]. 哈尔滨工程大学学报, 2016, 37(11): 1539-1545.
Liu Lei, Liu Feng, Lü Shuai, Ren Junqi. A formal description method of Dryad using membrane calculus[J]. Journal of Harbin Engineering University, 2016, 37(11): 1539-1545.

[13] Luo Mao, Li Chu-Min, Xiao Fan, Manyà Felip, Lü Zhipeng. An effective learnt clause minimization approach for CDCL SAT solvers [C]. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI 2017), Melbourne, Australia, August 19-25, 2017, 703-711.

[14] Matthew W Moskewicz, Conor F Madigan, Ying Zhao, Lintao Zhang, Sharad Malik. Chaff: Engineering an efficient SAT solver. [C]. In: Proceedings of the 38th Design Automation Conference (DAC 2001), Las Vegas, NV, USA, June 18-22, 2001, 530-535.

[15] Gilles Audemard, Laurent Simon. Refining restarts strategies for SAT and UNSAT. In: Proceedings of the 18th International Conference on Principles and Practice of Constraint Programming (CP 2012), Québec City, Canada, October 8-12, 2012, 118–126.