一阶逻辑定理证明器CSE中矛盾体分离式的简化方法

吴鑫, 陈树伟, 姜世攀

引用本文

吴鑫, 陈树伟, 姜世攀. 一阶逻辑定理证明器CSE中矛盾体分离式的简化方法[J]. 计算机科学, 2025, 52(5): 235-240.

WU Xin, CHEN Shuwei, JIANG Shipan. Simplification Method for Contradiction Separation Clause in First-order Logic AutomatedTheorem Prover CSE [J]. Computer Science, 2025, 52(5): 235-240.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.241000175

一阶逻辑定理证明器 CSE 中矛盾体分离式的简化方法

吴　鑫　　陈树伟　　姜世攀
西南交通大学数学学院　成都 611756
系统可信性自动验证国家地方联合工程实验室　成都 611756
(wuxin8@outlook.com)

摘　要　一阶逻辑自动定理证明器能够解决大量形式化后的实际问题,具有重要的应用价值。矛盾体分离演绎发展了自动定理证明领域经典的归结原理,具有更强的证明能力。在基于矛盾体分离规则的自动定理证明器 CSE(Contradiction Separation Extension)的基础上,提出一种矛盾体分离式的简化算法,通过优化数据结构,使用指针保存子句间的互补信息,并在此基础上选择实际参与演绎的子句,从而得到最简矛盾体分离式。这种新的简化算法可生成更简短的分离式,进一步利用子句的合一互补性,增强空子句演绎路径的检测能力,提高证明器的效率。实验结果显示,相比 CSE,使用此简化算法后的证明器 CSE_BSCS 能多证明 39 个测试例,平均证明时间减少了 18.64%,在证明能力和效率上均更优。

关键词:矛盾体分离;矛盾体分离式简化;最简矛盾体分离式;互补信息;证明器
中图分类号　TP181

Simplification Method for Contradiction Separation Clause in First-order Logic Automated Theorem Prover CSE

WU Xin,CHEN Shuwei and JIANG Shipan
School of Mathematics,Southwest Jiaotong University,Chengdu 611756,China
National-Local Joint Engineering Laboratory of System Credibility Automatic Verification,Chengdu 611756,China

Abstract　First-order logic automated theorem proving has the capacity to resolve a multitude of practical problems after formalization,and thus holds considerable practical value. As an advancement in automated theorem proving,contradiction separation deduction extends the classical resolution principle and exhibits enhanced proving capability. In this paper,a simplified algorithm for contradiction separation is proposed and theoretically proven based on the Contradiction Separation Extension(CSE) prover,which follows the rule of contradiction separation. The proposed algorithm enhances efficiency via data structure optimization,utilizing pointers to store complementary information between clauses. This information is then employed to select the clauses that are involved in deductions,thereby achieving the separation clause simplification. This approach produces shorter separation clauses while leveraging the unification complementarity of clauses to strengthen the detection capability of empty clause derivation paths,ultimately improving prover efficiency. Experimental results demonstrate that the enhanced prover CSE_BSCS with this simplification algorithm solves 39 additional test cases compared to the original CSE,with an 18.64% reduction in average proving time. These improvements confirm the superior performance of CSE_BSCS in both proving capability and efficiency over CSE.

Keywords　Contradiction separation,Contradiction separation simplification,Separation clause simplification,Complementary information,Prover

1　引言

自动推理是人工智能研究的分支之一,主要研究如何使用计算机帮助人们进行推理,其对人工智能推理的发展至关重要[1]。经典逻辑包含命题逻辑和一阶逻辑:命题逻辑中的原子公式即为最小单元;而一阶逻辑中的原子公式存在项结构,其表达能力更加丰富。许多实际问题被形式化为一阶逻辑问题后,可以使用一阶逻辑自动定理证明器进行验证。因此,一阶逻辑自动定理证明器具有巨大的应用潜力和重要的研究价值[2]。

推理方法是一阶逻辑自动定理证明器的核心技术。1965年,Robinson[3]建立了归结原理。在此之后,语义归结[4]、

到稿日期:2024-10-30　返修日期:2024-12-06
基金项目:国家自然科学基金(61976130)
This work was supported by the National Natural Science Foundation of China (61976130).
通信作者:陈树伟(swchen@swjtu.edu.cn)

线性归结[5]、锁归结[6]等更高效的归结方法相继出现，且被广泛用于自动定理证明器，如定理证明器 Vampire[7]、定理证明器 E[8]等。这些归结方法在一定程度上提高了演绎的效率，但核心仍然是二元归结，并不能使用多个子句进行演绎。2018 年，Xu 等[9]提出了允许多个子句参与并消除多组互补的矛盾体分离规则，并证明了二元归结是矛盾体分离规则的特例。随后，Cao 等[10]基于矛盾体分离规则，开发了一种自动定理证明器(CSE)。CSE 作为新兴的一阶逻辑自动定理证明器，理论十分先进，但由于矛盾体分离演绎的形式多种多样，因此需要设计高效的矛盾体分离演绎算法。

矛盾体分离演绎的核心技术在于如何构建标准矛盾体。Chen 等[11]在现有 CSE 中的标准矛盾体构建方式上，提出了按照决策文字策略的形式构建标准矛盾体，通过不同方式将控制子句加入标准矛盾体，从而形成不同的矛盾体分离演绎算法。Cao 等[12-13]进一步根据不同的矛盾体分离特点，提出的基于较优子句、基于优化演绎路径和基于充分使用子句的矛盾体分离演绎算法。这些算法能够快速演绎出子句，大幅度提高了证明器的证明效率，发挥了矛盾体分离的多元性和动态性，使得在矛盾体分离演绎过程中可以尽可能地搜索更多的路径。Liu 等[14]提出的基于充分使用二元子句的矛盾体分离演绎算法，将二元子句经过不同的合一替换加入标准矛盾体，充分发挥二元子句的演绎能力，从而提高 CSE 证明器的证明能力，并成功地将矛盾体分离演绎算法与定理证明器 Vampire[15]和定理证明器 E[16]结合。

此外，构建标准矛盾体时选择子句的方式也会对矛盾体分离演绎产生影响。Zeng 等[17]基于子句间的互补关系，定义基于决策文字的互补比，提出基于多子句动态标准矛盾分离推理规则的互补比算法，从而指导子句参与演绎和规划演绎的路径，提高了对困难问题的求解能力。Jiang 等[18]在矛盾体分离演绎的过程中，提出了基于单元子句中互补对的子句选择和基于互补对分布和演绎距离的多种文字和子句选择策略，其可以在一定程度上提升 CSE 的定理证明能力。

上述算法在构建标准矛盾体后，均会对参与演绎的子句进行选择。当前 CSE 按照子句加入标准矛盾体的先后顺序进行遍历，以子句中的决策文字在标准矛盾体中是否存在互补对为标准，选取实际参与演绎的子句。在实际应用中，上述方法选取的演绎子句过多，导致得到的矛盾体分离式并非最简矛盾体分离式，进而影响演绎效率。因此，研究最简矛盾体分离式的方法具有重要意义，能为提升证明器的证明能力和效率提供一种新的思路和方法。

本文通过改进 CSE 的数据结构，利用指针存储子句间的互补信息，根据互补信息选取实际参与演绎的子句，从而得到最简矛盾体分离式。该方法能够以更小的时间复杂度得到最简矛盾体分离式。

2 矛盾体分离规则

2.1 一阶逻辑

定义 1(子句和子句集[19]) 子句是由若干文字(原子公式或其否定形式)析取而成，由子句组成的集合被称为子句集。

定义 2(互补对[19]) 文字 p 及其否定 $\neg p$ 称为互补对。

定义 3(项[19]) 一阶逻辑中的项被递归定义为：常元符号是项，变元符号是项。若 f 是 n 元函数符号，$t_1, t_2, \cdots, t_n$ 是项，则 $f(t_1, t_2, \cdots, t_n)$ 是项。所有项都是由此递归方式有限次生成的。

定义 4(替换[19]) 在一阶逻辑中，若 $x_1, x_2, \cdots, x_n$ 是不同的变元，$t_1, t_2, \cdots, t_n$ 是项，且 $x_i \neq t_i (i=1,2,\cdots,n)$，则称 $\theta = t_1/x_1, t_2/x_2, \cdots, t_n/x_n$ 为替换。

2.2 矛盾体分离动态演绎

定义 5(标准矛盾体[9]) 假设一阶逻辑子句集 $S=\{C_1, C_2, \cdots, C_m\}$，对于任意 $(p_1, p_2, \cdots, p_m) \in \prod_{i=1}^{m} C_i$，$(p_1, p_2, \cdots, p_m)$ 至少存在一组互补对，则称 $S=\bigwedge_{i=1}^{m} C_i$ 为标准矛盾体。

定义 6(矛盾体分离规则[9]) 假设一阶逻辑子句集 $S=\{C_1, C_2, \cdots, C_m\}$。不失一般性，假设子句 $C_1, C_2, \cdots, C_m$ 之间不存在相同的变元(若存在相同的变元，则需做变元更名)，以下从子句集 S 产生新子句的推理规则被称为标准矛盾体分离规则(简称 S-CS 规则)。

对于任意 $C_i (i=1,\cdots,m)$，首先对其进行替换，将 σ_i 变为 $C_i^{\sigma_i}$，然后将子句分为两部分 $C_i^{\sigma_i-}$ 和 $C_i^{\sigma_i+}$，使得：

(1) $C_i^{\sigma_i} = C_i^{\sigma_i-} \vee C_i^{\sigma_i+}$，其中 $C_i^{\sigma_i-}$ 和 $C_i^{\sigma_i+}$ 没有相同文字；

(2) $C_i^{\sigma_i+}$ 可以为空子句，$C_i^{\sigma_i-}$ 不能为空子句；

(3) $\bigwedge_{i=1}^{m} C_i^{\sigma_i-}$ 是一个标准矛盾体。

其中，产生的子句 $\bigvee_{i=1}^{m} C_i^{\sigma_i+}$ 称为矛盾体分离式，$\bigwedge_{i=1}^{m} C_i^{\sigma_i-}$ 称为标准矛盾体。

定义 7(矛盾体分离动态演绎[9]) 假定 $S=\{C_1, C_2, \cdots, C_m\}$ 是一阶逻辑的一个子句集，如果 $\Phi_i (i=1,2,\cdots,t)$ 满足如下两个条件，则称 $\Phi_1, \Phi_2, \cdots, \Phi_t$ 为一个从 S 到 Φ_t 的矛盾体分离动态演绎(简称 S-CS 演绎)。

(1) $\Phi_i \in S, h=1,2,\cdots,t, t=1,2,\cdots$；

(2) 存在 $r_1, r_2, \cdots, r_{k_i} < i$，使得 $\Phi_i = C_{r_{k_i}}^{s\theta_i}(\Phi_{r_1}, \Phi_{r_2}, \cdots, \Phi_{r_{k_i}})$，其中 $\theta_i = \bigcup_{j=1}^{k_i} \sigma_j$，而 $C_{r_{k_i}}^{s\theta_i}(\Phi_{r_1}, \Phi_{r_2}, \cdots, \Phi_{r_{k_i}})$ 为矛盾体分离式。

定理 1(S-CS 演绎可靠性[9]) 矛盾体分离动态演绎的完备性：假定 $S=\{C_1, C_2, \cdots, C_m\}$ 是一阶逻辑的一个子句集，$\Phi_1, \Phi_2, \cdots, \Phi_t$ 为一个从 S 到 Φ_t 的矛盾体分离动态演绎，如果 Φ_t 是一个空子句 $\emptyset$，则子句集 S 为不可满足。

定理 2(S-CS 演绎可靠性[9]) 假定一阶逻辑子句集 $S=\{C_1, C_2, \cdots, C_m\}$，如果子句集 S 不可满足，则存在从 S 到空子句 $\emptyset$ 的基于标准矛盾体分离的动态演绎。

3 矛盾体分离式的简化方法

3.1 矛盾体分离式选取方法

目前，CSE 利用决策文字构建标准矛盾体[20]，即在演绎结束时，系统根据当前标准矛盾体中决策文字的合一互补情况选择实际参与演绎的子句，形成新的更小的标准矛盾体。若在矛盾体中，某个决策文字不存在互补文字，则需要在矛盾体和分离式中删除该子句的文字，其余子句仍按照矛盾体

分离演绎规则进行演绎。

矛盾体分离演绎的过程,实质上是标准矛盾体和矛盾体分离式的构建过程与决策文字集的更新过程。但在实际演绎过程中,特别是演绎前期,当一个新子句加入标准矛盾体时,需要生成包含该子句参与演绎的分离式(中间分离式)。若不进行演绎子句的选择,则得到的矛盾体分离式不是最简的,且演绎路径中会包含大量无关的子句,这些无关的子句会影响演绎的效率。

例 1　假设一阶逻辑子句集 $S=\{C_1,C_2,C_3,C_4,C_5,C_6\}$,其中 $C_1=P_1(a)$,$C_2=P_2(b)$,$C_3=\neg P_1(x_1)\vee P_3(x_2)$,$C_4=\neg P_2(x_3)\vee P_4(c)\vee P_5(a)$,$C_5=\neg P_4(x_3)\vee P_6(a)\vee P_7(b)$,$C_6=\neg P_1(x_1)\vee\neg P_2(x_3)\vee\neg P_3(x_2)$。对子句集进行替换,则有 $C_1^\sigma=P_1(a)$,$C_2^\sigma=P_2(b)$,$C_3^\sigma=\neg P_1(a)\vee P_3(b)$,$C_4^\sigma=\neg P_2(b)\vee P_4(c)\vee P_5(a)$,$C_5^\sigma=\neg P_4(c)\vee P_6(a)\vee P_7(b)$,$C_6^\sigma=\neg P_1(a)\vee\neg P_2(b)\vee\neg P_3(b)$。之后按照 C_1^σ,C_2^σ,C_3^σ,C_4^σ,C_5^σ,C_6^σ 的顺序构建标准矛盾体,决策文字集分别为:$D_{l_2}=\{P_1(a),P_2(b)\}$,$D_{l_3}=\{P_1(a),P_2(b),P_3(b)\}$,$D_{l_4}=\{P_1(a),P_2(b),P_3(b),P_4(c)\}$,$D_{l_5}=\{P_1(a),P_2(b),P_3(b),P_4(c),P_6(a)\}$。若不进行子句的选择,则矛盾体分离的演绎形式如表 1 所列。此时,矛盾体分离式为 $C_6^\sigma(C_1,C_2,C_3,C_4,C_5,C_6)=P_5(a)\vee P_7(b)$,标准矛盾体为 $(P_1(a))\wedge(P_2(b))\wedge(\neg P_1(a)\vee P_3(b))\wedge(\neg P_2(b)\vee P_4(c))\wedge(\neg P_2(b)\vee P_4(c))\wedge(\neg P_1(a)\vee\neg P_2(b)\vee\neg P_3(b))$。

表 1　不进行演绎子句选择的矛盾体分离演绎

Table 1　Contradiction separation deduction without deductive clause selection

	C^σ	$C^{\sigma+}$	$C^{\sigma+}$
C_1	$P_1(a)$		$P_1(a)$
C_2	$P_2(b)$		$P_2(b)$
C_3	$\neg P_1(a)\vee P_3(b)$		$\neg P_1(a)\vee P_3(b)$
C_4	$\neg P_2(b)\vee P_4(c)\vee P_5(a)$	$P_5(a)$	$\neg P_2(b)\vee P_4(c)$
C_5	$\neg P_4(c)\vee P_6(a)\vee P_7(b)$	$P_7(b)$	$\neg P_4(c)\vee P_6(a)$
C_6	$\neg P_1(a)\vee\neg P_2(b)\vee\neg P_3(b)$		$\neg P_1(a)\vee\neg P_2(b)\vee\neg P_3(b)$

不难发现,若从矛盾体和分离式中删除子句 C_5,其余的矛盾体 $\bigwedge_i C_i^{\sigma_i-}(i=1,2,3,4,6)$ 仍是标准矛盾体,且能得到更简短的分离式 $P_5(a)$。若进一步删除子句 C_5,其余的矛盾体 $\bigwedge_i C_i^{\sigma_i-}(i=1,2,3,6)$ 仍是标准矛盾体,其矛盾体分离式为 $\emptyset$。

在 CSE 证明器中,系统采用遍历决策文字集的方式删除子句,遍历顺序为决策文字加入决策文字集的顺序。假设决策文字集为 D_l,标准矛盾体对应的文字集合为 SC,矛盾体分离式为 CSC。算法流程如算法 1 所示。

算法 1　遍历决策文字集删除子句的矛盾体分离演绎算法

输入:决策文字集 D_l,标准矛盾体中的文字集合 SC;

输出:矛盾体分离式 CSC;

```
1. i=0;
2. while i<Dl. length do
3.     j= 0;
4.     取出Dl中第 i 个文字Pi;
5.     while j< SC. length do
6.       取出 SC 中第 j 个文字Qj;
7.       if Pi与Qj合一 then
8.         标记Pi所在的子句;
9.         i→i+1;
10.        break;
11.      else
12.        j→j+1;
13.      end if
14.    end while
15. end while
16. 使用标记的子句进行矛盾体分离演绎,并生成分离式 CSC。
```

算法 1 通过遍历标记子句找出决策文字在矛盾体中有互补文字的子句,然后根据标记结果进行矛盾体分离演绎。但在未标记的子句中,矛盾体中的决策文字不存在互补文字,而该子句在矛盾体中的其余文字可能与标记子句的决策文字互补,导致该标记的子句未被删除。

例 2　初始参与演绎的子句有 $\{C_1,C_2,C_3,C_4,C_6\}$,遍历决策文字集 $D_{l_5}=\{P_1(a),P_2(b),P_3(b),P_4(c),P_6(a)\}$ 后,能够选择出的标记子句有 $\{C_1,C_2,C_3,C_4,C_6\}$,对应的矛盾体分离的演绎形式如表 2 所列。此时,矛盾体分离式为 $C_5(C_1,C_2,C_3,C_4,C_6)=P_5(a)$,标准矛盾体为 $(P_1(a))\wedge(P_2(b))\wedge(\neg P_1(a)\vee P_3(b)\wedge(\neg P_2(b)\vee P_4(c))\wedge(\neg P_1(a)\vee\neg P_2(b)\vee\neg P_3(b))$。

表 2　遍历决策文字集的方式删除子句的矛盾体分离演绎

Table 2　Process of deductive contradiction separation clauses by traversing the way of the set of decision literals

	C^σ	$C^{\sigma+}$	$C^{\sigma-}$
C_1	$P_1(a)$		$P_1(a)$
C_2	$P_2(b)$		$P_2(b)$
C_3	$\neg P_1(a)\vee P_3(b)$		$\neg P_1(a)\vee P_3(b)$
C_4	$\neg P_2(b)\vee P_4(c)\vee P_5(a)$	$P_5(a)$	$\neg P_2(b)\vee P_4(c)$
C_6	$\neg P_1(a)\vee\neg P_2(b)\vee\neg P_3(b)$		$\neg P_1(a)\vee\neg P_2(b)\vee\neg P_3(b)$

经过算法 1 得到分离式 $P_5(a)$,其相比于例 1 中所述的分离式 $P_5(a)\vee P_7(b)$,所含文字数更少。但此时的标准矛盾体中不存在子句 C_4 对应的决策文字 $P_4(c)$ 的合一互补文字,再次选择后可得到新的矛盾体分离式 $C_4(C_1,C_2,C_3,C_6)=\emptyset$,标准矛盾体为 $(P_1(a))\wedge(P_2(b))\wedge(\neg P_1(a)\vee P_3(b))\wedge(\neg P_2(b)\vee P_4(c))\wedge(\neg P_1(a)\vee\neg P_2(b)\vee\neg P_3(b))$。

若子句未被标记,则系统不会将该子句的文字从标准矛盾体中删除,从而影响其余子句的标记判断。通过反向遍历决策文字集能够解决此问题。但在实际演绎中,系统需生成中间分离式,未标记子句的文字从标准矛盾体中删除后,在后续演绎时又需重新加入,此过程需多次对存放标准矛盾体的容器进行增删操作,影响了演绎效率。

3.2　矛盾体分离动态演绎

为了进一步提高演绎效率,本文从分离式的本质出发,提出了矛盾体分离式简化算法。分离式的本质是通过决策文字的合一互补性进行选择,且子句加入标准矛盾体需要互补判断,若将这两个合一互补判断合并为一个,则能够提高证明效率。因此,本文在 CSE 的文字抽象类中添加一个指针成员变量(初始化为空指针)。在非单元子句加入标准矛盾体时,若该子句的文字与决策文字合一互补,则让该文字新添加的指针指向决策文字。构建完成的标准矛盾体的网络结构如图 1

所示，其中每个子句 C 的 $C^{\sigma-}$ 部分都存储了互补信息，这样就可以将子句集划分为分离式和矛盾体。互补信息指向取决于新添加的指针。

图 1 中的左侧子句为最后加入标准矛盾体的子句，在选择演绎子句时，以最后加入的子句为起点，根据子句 $C^{\sigma-}$ 中存储的互补信息指向选取演绎子句。相应的矛盾体分离式简化算法如算法 2 所示。

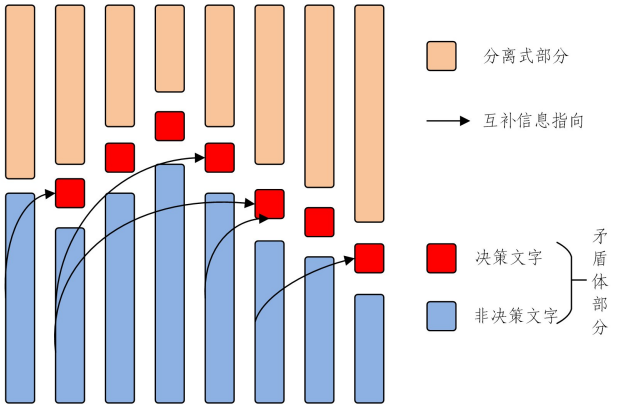


图 1 标准矛盾体网络结构图

Fig. 1 Standard contradiction network structure diagram

算法 2 矛盾体分离式的简化算法

输入：子句成功参与演绎的矛盾体网络结构

输出：子句 C_i 参与演绎的最少演绎子句列表 L′(初始子句 C_i)，矛盾体分离式 CSC

```
1. i=0;
2. while i<Ci. length do
3.     子句参与演绎的最少演绎子句列表 L′(初始子句Ci)，矛盾体分离式 CSC;
4.     i→i+1;
5.     计算子句Ci的文字个数 N;
6.     if N=1 then
7.         切换下一个子句;
8.     end if
9.     for j 从 1 到 N do
10.        if Pj∈SC then
11.            得到文字Pj指向的决策文字 P′;
12.            将决策文字 P′对应的子句加入 L′;/ * 此过程中子句会重复加入判断 * /
13.        else
14.            将文字Pj加入矛盾体分离式 CSC 中;
15.        end if
16.    end for
17. end while
```

在算法 2 中，将演绎子句列表 L' 初始化为 C_i，然后从子句 C_i 开始，根据文字存储的互补信息将下一个参与演绎的子句加入到子句列表 L' 中(第 10－13 行)，同时，将每个子句的 $C^{\sigma+}$ 加入到分离式中(第 14 行)。若访问到单元子句，则切换下一个子句继续查找演绎子句，直到演绎子句列表 L' 中的子句不再增加，且完全访问了演绎子句列表 L' 中的子句，算法结束，完成演绎子句的选择。

例 3 初始参与演绎的子句有 $\{C_1, C_2, C_3, C_4, C_6\}$，在每个子句加入标准矛盾体的文字(除决策文字)过程中，均包含指向自身互补决策文字的指针。根据算法 2，演绎子句列表 L' 初始化为 $\{C_6\}$，根据子句 C_6 的文字指针依次选出子句 C_1，C_2，C_3，然后依次访问子句 C_1，C_2，C_3 的文字指针信息。最后选出的实际参与演绎的子句有 $\{C_6, C_1, C_2, C_3\}$，对应矛盾体分离的演绎形式如表 3 所列。此时，矛盾体分离式 $C_4(C_1, C_2, C_3, C_6)=\varnothing$，标准矛盾体分离式为 $(P_1(a)) \wedge (P_2(b)) \wedge (\neg P_1(a) \vee P_3(b)) \wedge (\neg P_1(a) \vee P_3(b))$。

表 3 使用矛盾体分离式简化算法的矛盾体分离演绎

Table 3 Contradiction separation deduction using contradiction separator simplified algorithm

	C^{σ}	$C^{\sigma+}$	$C^{\sigma-}$
C_1	$P_1(a)$		$P_1(a)$
C_2	$P_2(b)$		$P_2(b)$
C_3	$\neg P_1(a) \vee P_3(b)$		$\neg P_1(a) \vee P_3(b)$
C_6	$\neg P_1(a) \vee \neg P_2(b) \vee \neg P_3(b)$		$\neg P_1(a) \vee \neg P_2(b) \vee \neg P_3(b)$

经过算法 2，可以得到分离式 $\varnothing$，该分离式显然比例 1 中所述分离式 $P_5(a) \vee P_7(b)$ 和例 2 中所述分离式 $P_5(a)$ 所含文字数更少。从例 3 可以看出，相比原演绎子句选取算法，矛盾体分离式的简化算法没有进行重复的文字合一判断，能够提高证明器的证明效率，且得到的分离式更加简短，有着更强的空子句演绎路径的检测能力。

系统在进行合一互补判断时，会从文字的谓词是否互补开始判断，然后递归判断文字的项结构是否合一相同。而合一互补判断是原子句选择算法中最耗时的操作。不同于原演绎子句选择算法，分离式简化算法不进行文字间的合一互补判断，而是直接根据指针信息寻找子句，并且按照此算法选择出的标准矛盾体与矛盾体分离式是最简的矛盾体与分离式。

命题 1 假设一阶逻辑子句集 $S=\{C_1, C_2, C_3, \cdots, C_n\}$，已加入 $i-1(1<i<n)$ 个子句 $\{C_1, C_2, \cdots, C_{i-1}\}$，决策文字集为 $D_{l_{i-1}}=\{d_m \mid d_m \in C_m, m=1, 2, \cdots, i-1\}$，现将子句 C_i 加入演绎子句集中，并生成标准矛盾体 $C_i=\bigwedge_{t=1}^{i} C_t^{\sigma_t-}$，矛盾体分离式为 $\bigvee_{t=1}^{i} C_t^{\sigma_t+}$。采用算法 2 对该矛盾体分离式进行简化，选择出的子句有 $\{C_{s_1}, C_{s_2}, \cdots, C_{s_k}, C_i\}(1 \leqslant s_1 < s_2 < \cdots < s_k < i)$，则子句 $C_j^{\sigma_j-}$ 的合取式 $C_{k+1}=\bigwedge_j C_j^{\sigma_j-}$ 仍为标准矛盾体，且 C_{k+1} 是包含 $C_i^{\sigma_i-}$ 且子句数量最少的标准矛盾体，其矛盾体分离式为 $\bigvee_j C_j^{\sigma_j+}$。

证明：首先证明 $C_{k+1}=\bigwedge_j C_j^{\sigma_j-}(j=s_1, s_2, \cdots, s_k, i, 1 \leqslant j \leqslant i)$ 是标准矛盾体。在对子句集 $\{C_{s_1}, C_{s_2}, \cdots, C_{s_k}, C_i\}$ 的分离中，$C_j^{\sigma_j}=C_j^{\sigma_j-} \vee C_j^{\sigma_j+}$，其中 $C_j^{\sigma_j-}$ 与 $C_j^{\sigma_j+}$ 不含相同的文字，$C_j^{\sigma_j-}$ 不为空，且 $\bigvee_j C_j^{\sigma_j+}=C_{k+1}^{\sigma}(C_{s_1}, C_{s_2}, \cdots, C_{s_k}, C_i)$，$\sigma=\bigcup_j \sigma_j$，故该分离方式满足矛盾体分离规则(定义 6)的条件(1)和(3)。

下面利用反证法证明该分离方式满足分离规则中的条件(2)，即任意 $(l_{s_1}, l_{s_2}, \cdots, l_{s_k}, l_i) \in \prod_j C_j^{\sigma_j-}$ 存在互补对。假设 $(l_{s_1}, l_{s_2}, \cdots, l_{s_k}, l_i) \in \prod_j C_j^{\sigma_j-}$ 不存在互补对，若子句 C_{s_1}，C_{s_2}，$\cdots$，C_{s_r} 为单元子句，则 $l_{n_1} \in \{d_{n_1}\}(n_1=s_1, s_2, \cdots, s_r)$ 全为决策文字。当 $l_{s_{r+1}}$ 不是决策文字时，即 $l_{s_{r+1}} \in C_{s_{r+1}}^{\sigma_{r+1}-}/d_{s_{r+1}}$，根

据子句成功加入标准矛盾体的条件，文字集$\{l_{s_1},l_{s_2},\cdots,l_{s_r}\}$必然存在与$l_{s_{r+1}}$合一互补的文字，这与$(l_{s_1},l_{s_2},\cdots,l_{s_k},l_i)$不存在互补对矛盾，故$l_{s_{r+1}}\in\{d_{s_{r+1}}\}$。同理，$l_{s_{s_2}}\in\{d_{s_{n_2}}\}$($n_2=r+2$，$r+3,\cdots,k$)，此时文字集$(l_{s_1},l_{s_2},\cdots,l_{s_k})$里面的文字均是决策文字，且包含$C_i^{\sigma_i-}$中的所有互补文字，故$(l_{s_1},l_{s_2},\cdots,l_{s_k},l_i)$必然存在互补对，这与假设矛盾，故任意$(l_1,l_2,\cdots,l_k,l_i)\in\prod_j C_j^{\sigma_j-}$均存在互补文字。因此，$C_{k+1}=\bigwedge_j C_j^{\sigma_j-}$($j=s_1,s_2,\cdots,s_k,i,1\leqslant j\leqslant i$)是标准矛盾体，矛盾体分离式为$\bigvee_j C_j^{\sigma_j+}$。

同理，利用反证法证明标准矛盾体C_{k+1}是包含$C_j^{\sigma_j-}$且子句数量最少的标准矛盾体。假设存在$C_k=\bigwedge_j C_j^{\sigma_j-}$($j=s_1,s_2,\cdots,\alpha-1,\alpha+1,\cdots,s_k,i,1\leqslant s_1\leqslant\alpha\leqslant s_k<i$)是标准矛盾体，由于子句$C_\alpha$是根据互补文字选取出来的，则必然存在这样一条子句关系链$\{C_\alpha,C_{a_1},C_{a_2},\cdots,C_{a_{n_3}},C_i\}$($s_1\leqslant\alpha=\alpha_0<\alpha_1<\alpha_2<\cdots<\alpha_{n_3}\leqslant s_k$)，使得$\{\neg d_{a_0}\in\frac{C_{a_1}^{\sigma_{a_1}-}}{d_{a_1}},\neg d_{a_1}\in\frac{C_{a_2}^{\sigma_{a_2}-}}{d_{a_2}},\cdots,\neg d_{a_{n_3-1}}\in\frac{C_{a_3}^{\sigma_{a_3}-}}{d_{a_{n_3}}},\neg d_{a_{n_3}}\in C_i^{\sigma_i-}\}$。构造有序对$(l_{s_1},l_{s_2},\cdots,l_{\alpha-1},l_{\alpha+1},\cdots,l_{s_k},l_i)\in\prod_j C_j^{\sigma_j-}$，当$j=\alpha_h$($h=1,2,\cdots,n_3$)时，$l_j=\neg d_{a_{h-1}}$；当$j\notin\{\alpha_1,\alpha_2,\cdots,\alpha_{n_3},i\}$时，$l_j=d_j$；当$j=i$时，$l_i=\neg d_{a_{n_3}}$。上述有序对$(l_{s_1},l_{s_2},\cdots,l_{\alpha-1},l_{\alpha+1},\cdots,l_{s_k},l_i)$中的文字均是决策文字或决策文字的否定，由于决策文字间不存在互补对，且两两不相同(在挑选新的决策文字时，若选取了原决策文字集中的文字，则会生成冗余子句，系统会将该子句的文字从矛盾体和分离式中清除)，故该有序对不存在互补对。因此$C_k=\bigwedge_j C_j^{\sigma_j-}$($j=s_1,s_2,\cdots,\alpha-1,\alpha+1,\cdots,s_k,s_1\leqslant\alpha\leqslant s_k,1\leqslant j\leqslant i$)不能构成标准矛盾体，这与假设矛盾。故$C_{k+1}=\bigwedge_j C_j^{\sigma_j-}$($j=s_1,s_2,\cdots,s_k,i$)是包含$C_i^{\sigma_i-}$且子句数量最少的标准矛盾体，其矛盾体分离式为$\bigvee_j C_j^{\sigma_j+}$。

综上所述，子句$C_j^{\sigma_j-}$($j=s_1,s_2,\cdots,s_k,i$)的合取式$C_{k+1}=\bigvee_j C_j^{\sigma_j-}$仍为标准矛盾体，且$C_{k+1}$是包含$C_i^{\sigma_i-}$且子句数量最少的标准矛盾体，其矛盾体分离式为$\bigvee_j C_j^{\sigma_j+}$。证毕。

命题 1 说明了算法 2 获得的标准矛盾体是包含子句$C_i^{\sigma_i-}$的最小矛盾体，从而说明其对应的矛盾体分离式$\bigvee_j C_j^{\sigma_j+}$($j=l_1,l_2,\cdots,l_k,i$)是最简分离式。

4　实验及结果分析

4.1　实验准备

实验的计算机环境为 Intel Core(TM) i7-4790@3.6 GHz 处理器和 16 GB 内存，运行 64 位 Ubuntu 15.04 操作系统。测试集使用 2019－2023 年 CASC[21](CADE ATP System Competition)中的 FOF 竞赛例，去重后共有 1 513 个测试例，每个测试例的判定时间限定为 300 s。CSE 表示使用原矛盾体分离式选取方法的一阶逻辑自动定理证明器，CSE_BSCS (CSEBased on Separation Clause Simplification)表示使用矛盾体分离式简化算法的证明器。

4.2　CSE_BSCS 判定情况

相较于 CSE，CSE_BSCS 得到的矛盾体分离式更加简短，证明能力更强。此外，由于两种证明器获取的中间分离式形式不相同，所以两者之间的演绎路径并非完全相同，故在同一个测试例中，CSE_BSCS 并不比 CSE 的证明时间短。表 4 列出了证明器 CSE 和 CSE_BSCS 在 2019－2023 年测试例中的求解结果，表中的平均用时是指对应证明器在两个证明器共同求解(即两个证明器都求解出的问题)的测试例上的平均求解时间。

表 4　CSE_BSCS 与 CSE 的测试结果的对比

Table 4　Comparison of results between CSE_BSCS and CSE

年份	共同求解个数	CSE		CSE_BSCS	
		单独求解个数	平均用时/s	单独求解个数	平均用时/s
2019	105	8	39.55	**32**	**31.56**
2020	136	21	31.68	19	**21.83**
2021	138	17	23.83	**25**	30.32
2022	125	12	31.45	**32**	32 40
2023	103	8	33.81	**20**	29.06
总数(1513)	358	44	35.24	**83**	**28.67**

由表 4 可得，在测试例结果中，CSE_BSCS 的证明能力比 CSE 更强，证明效率更高。例如，在 1 513 个测试例中，CSE_BSCS 比 CSE 多证明了 39 个测试例，且平均用时减少了 18.64%。CSE_BSCS 证明能力和证明效率的提升，主要是因为其获取的中间分离式更加简短，更符合空子句目标导向的演绎方式，且新型分离式选取算法对空子句演绎路径的检验能力更强。

此外，在 CSE_BSCS 单独求解的 83 个测试例和 CSE 单独求解的 44 个测试例中，CSE_BSCS 比 CSE 多求解 20 个常识推理(Commonsense Reasoning，CSR)类型的测试例，以及 15 个软件验证(Software Verification，SWV)、软件确认(Software Validation，SWW)类型的测试例。其中，CSR 类型的测试例是常识推理领域的定理，SWV 和 SWW 类型的测试例均是软件验证领域的定理。

图 2 给出了 CSE_BSCS 和 CSE 在所有测试例上的性能分析结果，横坐标为证明器求解的测试例个数，纵坐标为求解测试例消耗的时间。

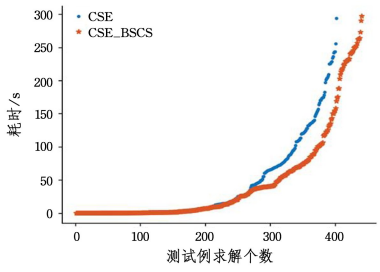


图 2　CSE_BICD 和 CSE 的性能对比

Fig. 2　Performance comparison of CSE_BICD and CSE

由图 2 可得，在前 20 s 内，CSE_BSCS 证明了 247 个定理，CSE 证明了 254 个定理；在 30s 内，CSE_BSCS 证明了 326 个定理，CSE 证明了 299 个定理；在 30 s 后，CSE_BSCS 证明的定理总数明显超过了 CSE；在 100 s 时，CSE_BSCS 证明的定理总数为 372，而 CSE 证明的定理总数为 344；在 200 s 时，CSE_BSCS 证明定理总数为 406，超出 CSE 证明定理总数 20 个；在 300 s 时，CSE_BSCS 证明定理总数为 441，超出 CSE 证明定理总数 39 个。

CSE_BSCS 使用了一种新的矛盾体分离式简化算法，能够更加充分地利用子句的合一互补性，优化了原中间分离式选取方法。在构建标准矛盾体时，通过指针存储文字的互补信息，根据指针的指向选取参与演绎的最少子句，从而选出简化的中间分离式。实验结果表明，CSE_BSCS 更有利于空子句演绎，具备更强的证明能力和更高的证明效率。

结束语 本文通过改进 CSE 的数据结构，利用指针存储子句间的互补信息，根据互补信息选取实际参与演绎的子句，得到最简矛盾体分离式，最后通过对照实验说明了改进算法的优越性。

子句在加入标准矛盾体时，会进行多次合一互补判断，且在算法实践中，回退机制虽然保证了子句的充分使用，但使得诸多子句多次尝试加入标准矛盾体，会损耗大量时间。本文提出的改进方法虽然能改善这种情况，但并不能彻底消除回退机制的弊端。在存储资源的限度下，平衡系统的空间复杂度和时间复杂度，设计更高效的矛盾体分离演绎框架，也是未来的重要研究内容。

参 考 文 献

[1] NDUNGI R, UYUN S. A review of automated reasoning and its applications in the 21st century[J]. The Indonesian Journal of Computer Science, 2023, 12(2): 483-491.

[2] CHEN G. Formal mathematics and proof engineering [J]. Newsletter of the Chinese Computer Society, 2016, 12(9): 40-44.

[3] ROBINSON J A. A machine-oriented logic based on the resolution principle[J]. Journal of the ACM, 1965, 12(1): 23-41.

[4] SLAGLE J R. Automatic theorem proving with renamable and semantic resolution[J]. Journal of the ACM, 1967, 14(4): 687-697.

[5] LOVELAND D W. A linear format for resolution[C]// Symposium on Automatic Demonstration. Berlin: Springer, 2006: 147-162.

[6] LIU X H. Automatic reasoning based on resolution method [M]// Beijing: Science Press, 1994: 15-107.

[7] KOVÁCS L, VORONKOV A. First-order theorem proving and Vampire[C] // International Conference on Computer Aided Verification. Berlin: Springer, 2013: 1-35.

[8] SCHULZ S, CRUANES S, VUKMIROVIĆ P. Faster, higher, stronger: E 2.3[C]// Automated Deduction-CADE 27: 27th International Conference on Automated Deduction. Natal, Brazil: Springer International Publishing, 2019: 495-507.

[9] XU Y, LIU J, CHEN S W, et al. Contradiction separation based dynamic multi-clause synergized automated deduction[J]. Information Sciences, 2018, 462: 93-113.

[10] CAO F. Research on an automated theorem prover for first-order logic based on contradiction separation [D]. Chengdu: Southwest Jiaotong University, 2020: 58-93.

[11] CHEN S W, XU Y, JIANG Y, et al. Some synergized clause selection strategies for contradiction separation based automated deduction[C]// 2017 12th International Conference on Intelligent Systems Knowledge Engineering(ISKE). IEEE, 2017: 1-6.

[12] CAO F, XU Y, CHEN S W, et al. A contradiction separation dynamic deduction algorithm based on optimized proof search[J]. International Journal of Computational Intelligence Systems, 2019, 12(2): 1245-1254.

[13] CAO F, XU Y, LIU J, et al. A multi-clause dynamic deduction algorithm based on standard contradiction separation rule[J]. Information Sciences, 2021, 566: 281-299.

[14] LIU P Y, WU G F, XU Y, et al. Extending e prover with fully use binary clauses algorithm based on standard contradiction separation rule[C]// 2021 16th International Conference on Intelligent Systems and Knowledge Engineering (ISKE). IEEE, 2021: 11-14.

[15] LIU P Y, XU Y, LIU J, et al. Fully reusing clause deduction algorithm based on standard contradiction separation rule[J]. Information Sciences, 2023, 622: 337-356.

[16] LIU P Y, CHEN S W, LIU J, et al. An efficient contradiction separation based automated deduction algorithm for enhancing reasoning capability[J]. Knowledge-Based Systems, 2023, 261: 110217.

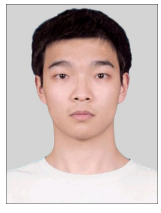
[17] ZENG G Y, CHEN S W, LIU J, et al. A complementary ratio based clause selection method for contradiction separation dynamic deduction [J]. Knowledge-Based Systems, 2024, 284: 111238.

[18] JIANG S P, CHEN S W. Clause and Literal Selection Strategies Based on Complementary Pair Distribution for Contradiction Separation Deduction[C]// International Conference on AI Logic and Applications. Singapore: Springer Nature Singapore, 2023: 214-226.

[19] ROBINSON A J, VORONKOV A. Handbook ofautomated reasoning[M]// Elsevier, 2001: 19-99.

[20] CAO F, WANG J, XU Y, et al. CSE-A Automated Theorem Prover Based on Standard Contradiction Separation Dynamic Deduction[J/OL]. https://www.researchsquare.com/article/rs-3955960/v1.

[21] SUTCLIFFE G, DESHARNAIS M. The 11th IJCAR automated theorem proving system competition-CASC-J11[J]. AI Communications, 2023, 36(2): 73-91.



WU Xin, born in 2001, postgraduate. His main research interests include automated reasoning and reinforcement learning.

CHEN Shuwei, born in 1977, Ph.D, associate professor. His main research interests include logic based automatedreasoning and decision analysis.

（责任编辑：何杨）