软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, [doi: 10.13328/j.cnki.jos.007128]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

舰载机弹药保障作业调度的形式化建模与验证 *

金钊[1,2,3], 金璐[1], 张博闻[4], 吴庆顺[1], 冯朔[1,2,3], 李冠峰[5], 徐明亮[1,2,3]

[1](郑州大学 计算机与人工智能学院,河南 郑州 450001)

[2](智能集群系统教育部工程研究中心,河南 郑州 450001)

[3](国家超级计算郑州中心,河南 郑州 450001)

[4](北京宇航系统工程研究所,北京 100076)

[5](中国船舶重工集团公司第七一三研究所,河南 郑州 450015)

通讯作者: 徐明亮, E-mail: iexumingliang@zzu.edu.cn

摘 要: 航母舰载机弹药保障作业的智能规划作为一种高效能航保作业调度方法,是助推航母工程先进技术建设发展的重要途径之一.高安全攸关属性下作业规划方案的正确性保证已经逐渐成为制约其实际应用部署安全的关键技术瓶颈.针对方案正确性验证中存在的弹药保障系统难建模、作业执行行为难描述、形式验证工具难实现等挑战,基于分离逻辑的思想,提出一种弹药保障系统的行为模型,并利用定理证明器 Coq 对作业规划方案进行形式化验证.首先提出一个符合弹药保障作业特征的序列化双层资源堆模型;基于该模型,构造一套可用于描述作业执行行为的建模语言及其操作语义;最后在 Coq 中实现一种证明辅助工具.通过几个典型弹药保障作业规划方案的交互式证明实例,验证工具的可用性与工程实用性.

关键词: 舰载机弹药保障作业;形式化验证;分离逻辑;操作语义;Coq

中图法分类号: TP311

中文引用格式: 金钊,金璐,张博闻,吴庆顺,冯朔,李冠峰,徐明亮.舰载机弹药保障作业调度的形式化建模与验证.软件学报. http://www.jos.org.cn/1000-9825/7128.htm

英文引用格式: Jin Z, Jin L, Zhang BW, Wu QS, Feng S, Li GF, Xu ML. Modeling and verification of carrier-borne aircraft ammunition support operation scheduling. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/7128.htm

Modeling and Verification of Carrier-Borne Aircraft Ammunition Support Operation Scheduling

JIN Zhao[1,2,3], JIN Lu[1], ZHANG Bo-Wen[4], WU Qing-Shun[1], FENG Shuo[1,2,3], LI Guan-Feng[5], XU Ming-Liang[1,2,3]

[1](School of Computer and Artificial Intelligence, Zhengzhou University, Zhengzhou 450001, China)

[2](Engineering Research Center of Intelligent Swarm Systems, Ministry of Education, Zhengzhou 450001, China)

[3](National Supercomputing Center in Zhengzhou, Zhengzhou 450001, China)

[4](Beijing Institute of Astronautical Systems Engineering, Beijing 100076, China)

[5](The 713 Research Institute of China Shipbuilding Industry Corporation, Zhengzhou 450015, China)

Abstract: An important way of boosting the development of advanced technology of aircraft carrier engineering is the intelligent planning of ammunition support operation of carrier aircraft, as an efficient scheduling method. Ensuring the correctness of operation planning schemes under the high safety critical attribute has gradually become the key technical bottleneck, restricting the security of their practical application deployments. Aiming at the challenges presented by the difficulties in modeling ammunition support systems, describing operation execution

* 基金项目: 国家自然科学基金重点项目(62036010); 国家自然科学基金杰出青年基金(62325602); 国家自然科学基金青年科学基金(62302459); 国家自然科学基金面上项目(61972362, 62372416)

收稿时间: 2023-09-11; 修改时间: 2023-10-30; 采用时间: 2023-12-13; jos 在线出版时间: 2024-01-05

behavior, and realizing formal verification tools, a behavior model is proposed for ammunition support systems based on the idea of separation logic, whereby the theorem prover Coq is used to formalize the operation planning scheme. First, a serialized two-tier resource heap model that conforms to the characteristics of ammunition support operation, is proposed. Subsequently, a set of modeling language and operational semantics are constructed based on this model, to describe job execution behavior. Finally, a proof assistant tool is implemented in Coq. The usability and engineering practicability of the tool are verified through interactive demonstrations of several typical ammunition support operation planning schemes.

Key words: carrier-borne aircraft ammunition support operation; formal verification; separation logic; operational semantics; Coq

航空母舰的战斗力核心指标是舰载机出动架次率,该指标由航空保障作业能力直接决定[1].舰载机弹药保障作业是耗时最长、任务最繁重的一项航空保障作业,需要一系列的弹药推车、电动叉车、起重设备、升降机、炸弹架、弹药库和机组人员的共同配合,构成一套复杂的舰载机航空弹药转运保障作业系统(简称弹药保障系统).若舰载机弹药保障作业出现问题或保障效率和舰载机战斗需求度不匹配,就会造成舰面保障作业周期延长,推迟舰载机出动时间,影响舰载机出动架次率和航母的综合作战能力[2].并且,弹药保障作业具有高安全攸关特性,作业过程中的任何安全风险都可能导致不可承受的灾难性结果[3].因此,弹药保障作业规划方案的正确性对于其实际应用部署与安全运作至关重要.

航母舰载机弹药保障作业调度是一个受时间、空间、资源约束的多任务复杂问题[4].自二战以来,以美军为代表,舰载机保障作业调度经历了从人工经验调度逐步转向计算机辅助调度,再到智能决策优化调度等三个阶段,其代表性的保障作业调度系统分别为占卜板(Ouija Board)、航空数据管理和控制系统(Aviation Data Management And Control System, ADMACS)和航母甲板作业规划决策支持系统(Deck Operations Course Of Action Planner, DCAP)[5].舰载机弹药保障作业调度常被抽象为柔性流水车间调度、多目标性、非线性指派等问题,采用智能优化、局部搜索、排队论等方法对其进行优化求解.但上述研究多以追求理论上的片面最优效用作为主要研究方向,忽略了作业规划方案的正确性,使得作业调度的可靠性难以得到保障.

弹药保障作业规划方案的正确性是指,对于上级委派的弹药转运保障作业任务,方案都能够将其依照既定的调度工序时序分配给弹药转运车,且满足航母资源约束和作业流程规范,并同时保证各转运车对方案整体和部分的转运都能有序送达正确作业位置.虽然现有研究所采用的系统仿真测试可以在一定程度上提升作业规划方案的可靠性[6],但由于仿真测试是一种面向实例的单个行为测试方法,故而逐一的测试是无法完全实现的,致使正确性难以得到保证.形式化验证[7]是一种提供可证保证的数学方法,它能够增加人们对计算系统按预期运行的信任,被广泛应用于安全攸关的硬件系统、通信协议、信息流以及分布式系统的正确性验证[8],以及航天器、飞机、高速列车、核反应堆等安全攸关领域的工业规模实际系统[9].该方法是面向性质的,使用逻辑推理的方法在论域中针对计算系统的所有行为给出一劳永逸的证明,从而具有准确、无误差、验证结果完备且可靠等优点.因此,本文采用形式化验证方法来保证弹药保障作业规划方案的正确性.

分离逻辑(separation logic)[10]方法是一种基于演绎推理的形式化验证方法,用于可耗竭资源分配与调度的模块化推理[11]及带有动态指针和内存数据结构的程序的正确性验证[12],并且在性质推理方面具有可组合性、模块性和可扩展性等优势.经过适当推广,分离逻辑被应用在更广泛的形式化验证领域,包括文件系统验证[13]、操作系统调度验证[14]、智能合约验证[15]、漏洞存在性验证[16]等.由于弹药保障作业资源调度与动态内存分配有相似之处,因此,分离逻辑可以在一定程度上满足作业规划方案正确性验证需求.另一方面,弹药保障系统兼具序列化和共享资源敏感特征,具体表现为:作业流程高度复杂、工序繁复精细并衔接紧凑;作业之间因资源的多机和多任务共享而存在大量复杂互斥或依赖关系.上述特征使得该系统在本质上不同于计算机软硬件、民航、物流等领域的计算系统,其所固有的复杂性导致应用分离逻辑方法难以进行系统行为建模.具体来说,弹药保障作业流程涉及多阶段保障位置和弹药转运车两类受限资源,且弹药本身也具有类型和转运优先级属性,但现有的分离逻辑模型多集中于计算机内存

单元这一种资源,尚不足以描述弹药保障系统的特征和保障资源调度行为的工序时序性.

为了提高形式化模型的实用性和验证效率,研究者一般会实现一个与模型配套的机械化验证工具,以实现在工业上的可用性和实用性.当前基于形式化验证的主流工具多基于定理证明,其又可细分为完全自动化的和交互式的.典型的完全自动化定理证明器是隶属于 Facebook 的 Infer[17].相对而言,交互式定理证明器又被称为“证明辅助工具”,其特点是无需为了实现自动化而牺牲验证系统和程序规范的表达能力,且无需依赖自动定理证明算法本身的正确性,因而验证结果更加可信.当前比较成熟的证明辅助工具有 Coq 和 Isabelle/HOL.其中 Coq 在基于分离逻辑的形式化验证中应用更广,典型的如 Cao 等人[18]开发的基于 Coq 的证明助手 VST-Floyd,该工具提供了一套半自动化策略来帮助用户构建 C 程序的功能正确性证明.与此同时,交互式定理证明研究也受到了国内学者的广泛关注.冯新宇等人[19]提出了一种用于验证抢占式操作系统调度的逻辑框架 OCAP 并实现了配套工具.Xu 等人[14]在 Coq 中实现了针对商业化嵌入式实时系统 uC/OS-II 的验证工具.詹博华[20]在 Isabelle 中开发了支持用户添加启发式搜索策略的自动推理工具 Auto2.章乐平等人[21]在 Isabelle 中形式化建模和验证了 L4 虚拟内存子系统.王小兵等人[22]在 Coq 中实现了一个支持索引式的 PPTL 验证工具.李亚男等人[23]在 Coq 中形式化建模与验证了 Lamport 的 Basic Paxos 算法.苏婉昀等人[24]研发了一种能够对动态数据结构的形状性质与数据约束进行融合推理的分离逻辑求解器 COMPSPEN.本文在前期工作[25]中基于 Coq 实现了一种基于分离逻辑的块云存储系统验证工具.总的来说,当前已有许多成熟的交互式定理证明工具,这些工具也已被成功应用于关键软件系统的验证.但据我们所知,目前尚未见专门针对共享资源敏感型作业规划任务方案的定理证明工具.

综上所述,为了保证舰载机弹药保障作业规划方案的正确性,本文基于分离逻辑的思想构建可用于方案正确性验证的形式化验证框架,并围绕弹药保障系统形式化建模、系统行为描述语言构建、交互式验证工具实现等问题展开研究,首先,构建一个序列化双层资源堆形式化模型;然后,提出一套专用于作业执行行为描述的建模语言 $\mathbf{ML}_{\mathbf{ASS}}$ 及其操作语义;最后,在 Coq 中实现一个与上述形式语义模型配套的证明辅助工具,并对几个典型的弹药保障作业规划方案进行交互式证明,以此来展示工具的可用性.基于此验证框架,可以通过使用符号化语言形式化地描述系统的行为,并以严格的语义解释为基础分析和验证方案,发现具有隐匿性、累积性的安全性资源使用冲突,揭示作业的行为机制,为安全攸关的弹药保障作业调度的可靠性保障提供理论基础和方法支撑.

本文第 1 节简单介绍弹药保障作业流程,然后给出弹药保障系统的形式化模型.第 2 节给出建模语言 $\mathbf{ML}_{\mathbf{ASS}}$ 的语法和语义.第 3 节给出基于 Coq 的弹药保障作业规划方案正确性的证明辅助工具,并引入方案的交互式证明实例.第 4 节给出总结和展望.

1 弹药保障系统建模

如图 1 所示,参照美“福特级”等航母的作业流程[26],舰载机弹药保障作业流程可以分为以下 5 个基本阶段:1)弹药出库;2)下层垂直转运;3)弹药装配;4)上层水平转运;5)舰面转运.该作业由大量保障人员(人)、舰载机(机)、保障车辆(车)构成的大规模“人-机-车”运动集群协作实施.

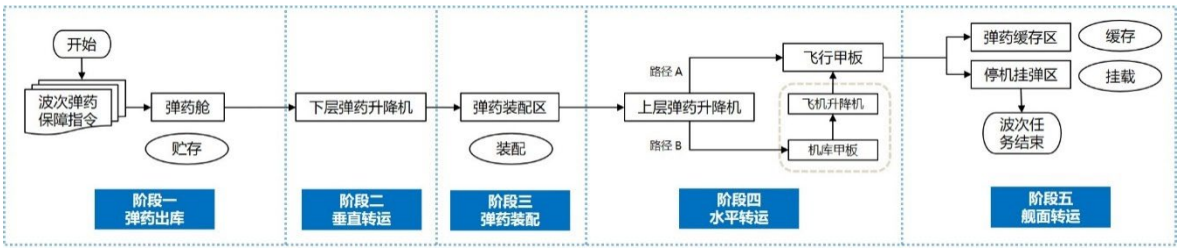

图 1 弹药保障作业流程示意图(非密)

现实中,弹药转运与保障任务会被依照作业规划方案既定的调度工序时序分配给转运车,各转运车依照作业流程次序实施转运,形成“方案到车”和“车到位置”所构成的一对多和多对多的架构关系.基于此,在作者前期工作[27]所提出的序列分离逻辑双层堆模型的基础上,本文将弹药保障系统的行为建模为一种两层分离的序列化双层资源堆结构模型.特别地,本模型细化了带类型弹药转运车所承担的子作业任务,并且对应实际情况,不同时序上的弹药转运车之间是完全独立的.而且能够描述每个保障位置下所进行保障作业的时长.

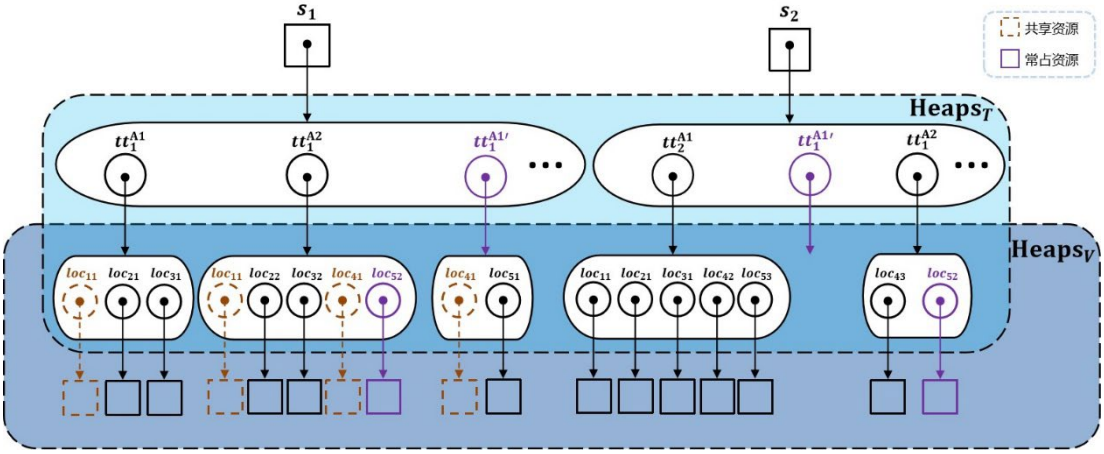

图 2 弹药保障系统的双层资源堆模型示意图

图 2 为本文设计的弹药保障系统的双层资源堆模型的示意图.该模型的核心组件是“车资源堆”(即图中的 $Heaps_T$)和“位置资源堆”(即图中的 $Heaps_V$).车资源堆用于搭建“方案到车的映射关系”和“车到目标位置的映射关系”之间的桥梁,并同时确保车资源之间的相互独立性.位置资源堆用于刻画转运任务底层的作业方式,其中的每个位置均表示一个共享可调配资源.可以看出,该结构能够清晰地刻画两种保障资源间的相互联系,即 $Heaps_V$ 中的某些位置从属于 $Heaps_T$ 中的某个转运车,如位置 loc_{32} 从属于 A2 类型弹药转运车 tt_1^{A2} .方案在弹药保障模型中的部署方式如下.方案 s_1 先被映射为一个由不同工序时序上的转运车 tt_1^{A1} 、tt_1^{A2} 和 $tt_1^{A1\prime}$ 所构成的序列,其中 tt_1^{A1} 和 $tt_1^{A1\prime}$ 分别代表 A1 类型弹药转运车从时序上划分的两个转运阶段.然后, $Heaps_T$ 将以上每个转运车分别映射为一个由不同作业次序上的位置所构成的序列.最后, $Heaps_V$ 将每个位置映射为保障位置下所需的作业时长.根据栈的后进先出的原则,即可描述方案 s_1 的实施过程.注意到在此方案中, A1 和 A2 型转运车共享了保障位置 loc_{11} 和 loc_{41} ,依照任务要求在 loc_{11} 处 A1 车的时序优先级更高,而在 loc_{41} 处 A2 车更高.对此,将 A1 车拆分成两个转运阶段,即可满足规划方案既定的工序时序性.此外,本结构还能够描述资源泄露情形.例如图中紫色部分代表常占资源,此类资源将无法被“重部署”.

综上所述,相比于现有的面向计算机内存系统的分离逻辑单风格堆模型,本文的序列化双层资源堆模型是符合弹药保障作业特征的工程模型,它的主要区别是将资源堆划分成了相互联系的两个部分:车资源堆映射($Heaps_T$)和位置资源堆映射($Heaps_V$).其中每个方案都由 $Heaps_T$ 中的一个车资源序列表示,而每个车都包含一个在 $Heaps_V$ 中有内容值的位置序列.从而,该模型具有描述作业规划方案的部署与实施的能力,可作为方案正确性验证的基础.

2 弹药保障作业描述语言 $\mathbf{ML_{ASS}}$

以第 1 节中的双层资源堆模型为基础,本文提出一种用于描述弹药保障系统中作业执行行为的建模语言,称作 $\mathbf{ML_{ASS}}$.首先定义该语言的语法,然后给出其形式语义.

2.1 $\mathbf{ML}_{\mathbf{ASS}}$ 的语法

首先给出用于描述 $\mathbf{ML}_{\mathbf{ASS}}$ 语法的记号的定义,如表 1 中所示.

表 1 $\mathbf{ML}_{\mathbf{ASS}}$ 的语法记号定义

– 表达式: 算术($e\in\langle\text{Exp}\rangle$)、布尔($be\in\langle\text{BExp}\rangle$)、规划方案($se\in\langle\text{SExp}\rangle$)、转运车($te\in\langle\text{TExp}\rangle$)
– $n,m\in\mathbb{Z}$: 整型常量; true,false: 布尔常量; **null**,**fin** $\in$ Atoms: 保留字
– $x,y\in\text{Var}$: 普通变量; $s_1,s_2\in\text{SVar}$: 方案变量; $t_1^a,t_2^a\in\text{TVar}$: 转运车变量

将 $\mathbf{ML}_{\mathbf{ASS}}$ 的表达式分为(算术)位置表达式(e)、布尔表达式(be)、方案表达式(se)和车表达式(te)四类.将 $\mathbf{ML}_{\mathbf{ASS}}$ 的命令集(C)分为基础命令、方案命令和车命令三类.基础命令是标准的,包括跳转(**skip**)命令、简单赋值命令、顺序、条件和 **while** 循环命令.方案命令包括方案规划、车工序附加和方案完成命令.车命令包括车工序规划、车工序增添、作业时长查询、车资源更新、首项作业执行和车资源移除命令.形式化地,可以将 $\mathbf{ML}_{\mathbf{ASS}}$ 的表达式和命令的语法以 BNF 的形式定义,如表 2 中所示.

表 2 $\mathbf{ML}_{\mathbf{ASS}}$ 的表达式和命令的语法定义

e	$::= n,m,... \mid x,y,... \mid e_1+e_2 \mid e_1-e_2 \mid e_1\times e_2 \mid \#te \mid \#s$
be	$::= e_1=e_2 \mid e_1\le e_2 \mid \textbf{true} \mid \textbf{false} \mid \neg be \mid be_1\wedge be_2 \mid be_1\vee be_2$
se	$::= \textbf{null} \mid \textbf{fin} \mid s_1,s_2,... \mid se\cdot te \mid se_1\cdot se_2$
te	$::= \textbf{null} \mid n,m,... \mid t_1^a,t_2^a,... \mid s.e$
C	$::= \textbf{skip} \mid x:=e \mid C;C' \mid \textbf{if}\ be\ \textbf{then}\ C\ \textbf{else}\ C' \mid \textbf{while}\ be\ \textbf{do}\ C'$
	$\mid s:=\textbf{asgn}\ (te^*) \mid \textbf{att}\ (s,te^*) \mid \textbf{comp}\ s$
	$\mid t^a:=\textbf{plan}\ (\overline{e}) \mid \textbf{add}\ (t^a,e) \mid x:=\{te.e\} \mid s.e:=te \mid \textbf{1stexe}\ s.e \mid \textbf{free}\ s.e$
其中 $\overline{e}$ 为位置表达式序列 $e_1,...,e_n$ 的简写, te^* 为车表达式序列 $te_1,...,te_n$ 的简写	

如表 2 中所述, $\mathbf{ML}_{\mathbf{ASS}}$ 的位置、方案和车表达式除包含各自类型的常量和变量作为基本单元以外,还引入了用于建立不同类型表达式之间联系的元素.具体地, $\#te$ 表示的是车 te 所承担转运目标位置的数目; $\#s$ 表示的是方案 s 所分配转运车的数目; $se\cdot te$ 表示的是由 se 所确定的车资源序列与 te 所确定的车资源的连接; $se_1\cdot se_2$ 的含义是类似的; $s.e$ 表示的是对应于方案 s 中第 i 项工序的车,其中 i 为位置表达式 e 的求值.

2.2 $\mathbf{ML}_{\mathbf{ASS}}$ 的语义

首先给出 $\mathbf{ML}_{\mathbf{ASS}}$ 的定义域.弹药保障系统模型可由以下五个组件来形式化: Stores_S 为从方案变量到车资源序列的映射; Stores_T 为从车变量到车资源的映射; Stores_V 为从位置变量到位置资源的映射; Heaps_T 为从车资源到位置资源序列的有限映射,可以使用间接资源查询记号 $\{te.e\}$ 对其进行访问,其中 te 代表所要访问的车, e 是车工序内部位置资源查询过程中特定位置的标识符; Heaps_V 为从位置资源到位置下作业时长值的映射.为了支持不受限的资源运算,本文将所有的值、位置资源和车资源都设置为整数.并对位置资源集合和车资源集合元素的取值进行设置,以确保资源分配总能成功.具体地,对于车资源集合 TT 和位置资源集合 Loc ,要求 TT 中有无穷多个长度为 m 的离散整数序列,即 $\text{TT}=\{tt_1,...,tt_m\}$;并要求 Loc 中有无穷多个长度为 n 的连续整数序列,即 $\text{Loc}=\{loc_1,...,loc_n\}$.并且,约定 TT 和 Loc 中的元素均为非负整数, ATom 中的元素取为负整数,从而使得保留字与资源的取值互不干扰.此外,可以通过引入指标集来形式化带有弹药类型的车.具体地,设 ATypes 是一个保留字集合的子集,其中的元素表示弹药类型标号.若对于任意的 $a\in\text{ATypes}$,存在唯一的对象 tt^a (tt^a 本身可能是集合),此时,定义以下集合

$$\mathcal{T}\triangleq\{tt^a\mid a\in\text{ATypes},\ tt\in\text{TT}\}. \tag{1}$$

则称 $\mathcal{T}$ 是以 ATypes 为指标集的集合.

由此,将带弹药类型的车资源序列的集合定义为

$$\mathcal{T}^*\triangleq\{(tt_1^a,...,tt_n^a)\mid tt_i^a\in\mathcal{T},n\in\mathbb{N}\}. \tag{2}$$

并且,对于任何 $\mathcal{T}^*$ 中的元素 $(tt_1^a,...,tt_n^a)$,以 $|(tt_1^a,...,tt_n^a)|$ 表示其长度,即 $|(tt_1^a,...,tt_n^a)|=n$.

另外,将位置资源序列的集合定义为

$$\mathrm{Loc}^* \triangleq \{(loc_1,...,loc_n) \mid loc \in \mathrm{Loc}, n \in \mathbb{N}\}. \tag{3}$$

并且,对于任何 Loc^* 中的元素 $(loc_1,...,loc_n)$,以 $|(loc_1,...,loc_n)|$ 表示其长度,即 $|(loc_1,...,loc_n)|=n$.

基于此,设 $f:X\to Y$ 表示 f 是从集合 X 到集合 Y 的完全函数,$f:X \rightharpoonup_{\mathrm{fin}} Y$ 表示 f 是从集合 X 到集合 Y 的有限部分函数.假设集合 $\mathcal{T}$、Loc 和 ATOms 是互不相交的.可以给出 $\mathbf{ML}_{\mathbf{ASS}}$ 的定义域的定义,如表 3 中所示.

表 3 $\mathbf{ML}_{\mathbf{ASS}}$ 的定义域的定义

$\mathrm{Values} \triangleq \mathbb{Z}$ $\mathrm{ATypes} \triangleq \{\mathrm{A1},\mathrm{A2},\mathrm{A3},...\}$ $\mathrm{SVar} \triangleq \{s_1,s_2,...\}$ $\mathrm{TVar} \triangleq \{t_1^a,t_2^a,...\}$
$\mathrm{Var} \triangleq \{x,y,...\}$ $\{\mathbf{null},\mathbf{fin}\}$, $\mathrm{ATypes} \subseteq \mathrm{Atoms}$ $\mathcal{T}$, Loc, $\mathrm{Atoms} \subseteq \mathrm{Values}$
$\mathrm{Stores}_S \triangleq \mathrm{SVar} \to \mathcal{T}^*$ $\mathrm{Stores}_T \triangleq \mathrm{TVar} \to \mathcal{T}$ $\mathrm{Stores}_V \triangleq \mathrm{Var} \to \mathrm{Values}$
$\mathrm{Heaps}_T \triangleq \mathcal{T} \rightharpoonup_{\mathrm{fin}} \mathrm{Loc}^*$ $\mathrm{Heaps}_V \triangleq \mathrm{Loc} \rightharpoonup_{\mathrm{fin}} \mathrm{Values}$

根据表 3 中的定义,将 $\mathbf{ML}_{\mathbf{ASS}}$ 的状态空间定义为

$$\mathrm{States} \triangleq \mathrm{Stores}_S \times \mathrm{Stores}_T \times \mathrm{Stores}_V \times \mathrm{Heaps}_T \times \mathrm{Heaps}_V. \tag{4}$$

根据式(4),一个弹药保障系统计算状态 $\sigma \in \mathrm{States}$ 为以下五元组

$$\sigma = (s_S, s_T, s_V, h_T, h_V). \tag{5}$$

以上述形式化为基础,可以给出 $\mathbf{ML}_{\mathbf{ASS}}$ 的语义.首先给出 $\mathbf{ML}_{\mathbf{ASS}}$ 的表达式的语义.通常,某个表达式的求值不依赖于所有的储存和堆.例如,位置表达式 $\#s$ 的求值只涉及方案储存 Stores_S.此外,在车表达式 $s.e$ 的求值中,如果 e 的求值结果 i 大于与方案 s 相关联车资源序列的长度,那么相应的语义条件将不会被满足.为处理此类情形,定义记号 *abort* 来表示潜在的类型错误,以确保所有的表达式均为良定义的.形式化地,可以通过使用结构归纳法给出 $\mathbf{ML}_{\mathbf{ASS}}$ 的表达式的指称语义定义,如表 4 中所示.

表 4 $\mathbf{ML}_{\mathbf{ASS}}$ 的表达式的指称语义定义

$[\![\#s]\!]\sigma = \lvert s_S(s)\rvert$ $[\![s]\!]\sigma = s_S(s)$ $[\![t^a]\!]\sigma = s_T(t^a)$
$[\![\mathbf{null}]\!]\sigma = \mathbf{null}$ $[\![\mathbf{fin}]\!]\sigma = \mathbf{fin}$ $[\![a]\!]\sigma = a$, 若 $a \in \mathrm{Atypes}$
$[\![s.e]\!]\sigma = tt_i^a$, 若 $s_S(s) = (tt_1^a,...,tt_n^a)$, $[\![e]\!]\sigma = i$, $[\![a]\!]\sigma = a$, 且 $1 \le i \le n$
$[\![s.e]\!]\sigma = abort$, 若 $s_S(s) = (tt_1^a,...,tt_n^a)$, $[\![e]\!]\sigma = i$, $[\![a]\!]\sigma = a$, 且 $n+1 \le i$
$[\![se.te]\!]\sigma = (tt_1^a,...,tt_n^a,tt^{a\prime})$, 若 $[\![se]\!]\sigma = (tt_1^a,...,tt_n^a)$ 且 $[\![te]\!]\sigma = tt^{a\prime}$
$[\![\#te]\!]\sigma = \begin{cases} k & \text{若 } h_T([\![te]\!]\sigma) = (loc_1,...,loc_k) \\ 0 & \text{若 } h_T([\![te]\!]\sigma) = \mathbf{null} \end{cases}$

在表 4 中,“**fin**”代表已实施的方案,“**null**”代表空序列,“$\cdot$”代表序列的连接运算符,从而,“**null**”是连接运算符“$\cdot$”的单位元.

接下来定义几种用于刻画双层资源堆结构的记号,如表 5 中所示.

表 5 用于刻画双层堆的记号的定义

– $\mathrm{dom}(h_H)$ 表示一个堆 $h_H \in \mathrm{Heaps}_H$ 的定义域,其中 h_H 论及 h_T 和 h_V, Heaps_H 论及 Heaps_T 和 Heaps_V
– $h_H \# h_H'$ 表示 h_H 和 h_H' 的定义域互斥
– $h_H * h_H'$ 表示一个取自于 h_H 和 h_H' 的联合的有限函数,并且要求 $h_H \# h_H'$ 成立
– $[f \mid x:a]$ 表示将 f 中的变量 x 映射为值 a,且将 f 的定义域 $\mathrm{dom}(f)$ 中的所有其他参数 y 映射为 $f(y)$
– $[f \upharpoonright s]$ 表示将 $\mathrm{dom}(f)$ 更新为定义域 s
– $a \in^{SEQ} (x_1,...,x_n)$ 表示元素 a 是序列 $(x_1,...,x_n)$ 的项
– $a_i \in_i^{SEQ} (x_1,...,x_n)$ 表示元素 a_i 是序列 $(x_1,...,x_n)$ 的第 i 项

下面给出 $\mathbf{ML}_{\mathbf{ASS}}$ 的命令的语义.可以采用小步操作语义法来定义,即通过定义格局之间的转换关系“$\rightsquigarrow$”来解

释命令.其中,格局包含状态 σ 、命令-状态序偶 $\langle C,\sigma\rangle$ (表示在状态 σ 下要执行命令 C)以及一种用以表示资源调度冲突的特殊格局 *stuck* .关系 $\rightsquigarrow$ 定义为

$$\langle C,\sigma\rangle \rightsquigarrow \sigma'. \tag{6}$$

它表示在状态 σ 下执行完命令 C 之后终止于终态 σ' .

基于此,用结构归纳法将 $\mathbf{ML_{ASS}}$ 的方案和车命令的操作语义定义如下.

方案规划命令:

$$\frac{[\![te_i]\!]\sigma \in \mathrm{dom}(h_T)(1\le i\le n)}{\langle s:=\mathbf{asgn}(te^*),\sigma\rangle \rightsquigarrow ([s_S \mid s:([\![te_1]\!]\sigma,...,[\![te_n]\!]\sigma)],s_T,s_V,h_T,h_V).} \tag{7}$$

车工序附加命令:

$$\frac{[\![te_i]\!]\sigma \in \mathrm{dom}(h_T)(1\le i\le n)且若tt^a \in^{SEQ} [\![s]\!]\sigma则tt^a \in \mathrm{dom}(h_T)}{\langle \mathbf{att}(s,te^*),\sigma\rangle \rightsquigarrow ([s_S \mid s:(s_S(s)\cdot([\![te_1]\!]\sigma,...,[\![te_n]\!]\sigma))],s_T,s_V,h_T,h_V).} \tag{8}$$

方案完成命令:

$$\frac{[\![s]\!]\sigma=\mathbf{null}}{\langle \mathbf{comp}\ s,\sigma\rangle \rightsquigarrow ([s_S \mid s:\mathbf{fin}],s_T,s_V,h_T,h_V).} \tag{9}$$

车工序规划命令:

$$\frac{tt^a \in \mathcal{T}-\mathrm{dom}(h_T)且loc_1,...,loc_n \in \mathrm{Loc}-\mathrm{dom}(h_V)}{\langle t^a:=\mathbf{plan}(\overline{e}),\sigma\rangle \rightsquigarrow (s_S,[s_T \mid t^a:tt^a],s_V,[h_T \mid tt^a:(loc_1,...,loc_n)],[h_V \mid loc_1:[\![e_1]\!]\sigma,...,loc_n:[\![e_n]\!]\sigma]).} \tag{10}$$

车工序增添命令:

$$\frac{h_T([\![t^a]\!]\sigma)=(loc_1,...,loc_n),loc_1,...,loc_n \in \mathrm{dom}(h_V),且loc_{n+1} \in \mathrm{Loc}-\mathrm{dom}(h_V)}{\langle \mathbf{add}(t^a,e),\sigma\rangle \rightsquigarrow (s_S,s_T,s_V,[h_T \mid [\![t^a]\!]\sigma:(loc_1,...,loc_n,loc_{n+1})],[h_V \mid loc_{n+1}:[\![e]\!]\sigma]).} \tag{11}$$

作业时长查询命令:

$$\frac{[e]\sigma=i,若loc \in^{SEQ} h_T([\![te]\!]\sigma)则loc \in \mathrm{dom}(h_V),且loc_i \in_i^{SEQ} h_T([\![te]\!]\sigma)}{\langle x:=\{te.e\},\sigma\rangle \rightsquigarrow (s_S,s_T,[s_V \mid x:h_V(loc_i)],h_T,h_V).} \tag{12}$$

车资源更新命令:

$$\frac{[\![s]\!]\sigma=(tt_1^a,...,tt_i^a,...,tt_n^a),[\![e]\!]\sigma=i(1\le i\le n),且tt_1^a,...,tt_n^a \in \mathrm{dom}(h_T)}{\langle s.e:=te,\sigma\rangle \rightsquigarrow ([s_S \mid s:(tt_1^a,...,[\![te]\!]\sigma,...,tt_n^a),s_T,s_V,h_T,h_V).} \tag{13}$$

首项作业执行命令:

$$\frac{\begin{array}{c}h_T(tt_i^a)=(loc_1,loc_2...,loc_k),loc_1,...,loc_k \in \mathrm{dom}(h_V),\\ [e]\sigma=i,tt_i^a \in_i^{SEQ} [\![s]\!]\sigma\ 且若tt^a \in^{SEQ} [\![s]\!]\sigma则tt^a \in \mathrm{dom}(h_T)\end{array}}{\langle \mathbf{1stexe}\ s.e,\sigma\rangle \rightsquigarrow (s_S,s_T,s_V,[h_T \mid [\![s.e]\!]\sigma:(loc_2,...,loc_k)],h_V\restriction(\mathrm{dom}(h_V)-\{loc_1\})).} \tag{14}$$

车资源移除命令:

$$\frac{\begin{array}{c}s_S(s)=(tt_1^a,...,tt_{k-1}^a,tt_k^a,tt_{k+1}^a,...,tt_n^a),\ tt_1^a,...,tt_n^a \in \mathrm{dom}(h_T),\ [\![s.e]\!]\sigma=tt_k^a\ (1\le k\le n),\\ h_T(tt_k^a)=\mathbf{null},\ 且对任意s',tt_k^a \notin^{SEQ} s_S(s')\end{array}}{\langle \mathbf{free}\ s.e,\sigma\rangle \rightsquigarrow ([s_S \mid s:(tt_1^a,...,tt_{k-1}^a,tt_{k+1}^a,...,tt_n^a)],s_T,s_V,h_T\restriction(\mathrm{dom}(h_T)-\{tt_k^a\}),h_V).} \tag{15}$$

下面对上述语义进行简要描述.首先,前三条方案命令操作 s_S . $s:=\mathbf{asgn}(te^*)$ 部署一个方案 s ,该方案被关联到由 te^* 表示的一组互不相同的带弹药类型的车资源构成的时序性序列,其中的每个序列项代表方案的一道车工序. $\mathbf{att}(s,te^*)$ 将 te^* 表示的一组车工序附加到现存方案 s 的末端.以上两条方案命令的语义均设有前提条件,要求无论是与现存方案关联的车资源,还是新的有待关联的车资源,都需要是在 h_T 中已定义完全的. $\mathbf{comp}\ s$ 将内容为空的方案 s ,设定为 **fin** ,表明此"被清空"的方案已实施完成.接下来的车命令访问或操作两种资源堆. $t^a:=\mathbf{plan}(\overline{e})$ 规划车

t^a 所承担的工序,依照转运流程次序分配一个位置序列,且各个位置对应的作业时长值构成序列 $\overline{e}$. **add** (t^a,e) 将在车 t^a 现有工序内容的末端添加保障时长为 e 的一项作业. $x:=\{te.e\}$ 读取车 te 的工序中第 i 个阶段位置所需的作业时长值,其中 i 为表达式 e 的求值. $s.e:=te$ 将方案 s 关联到的特定项车工序更新为 te . **1stexe** $s.e$ 将执行关联到方案 s 的特定项车工序的第一项弹药保障作业,在 h_T 中移除该车资源所对应保障位置序列的第一项,并在 h_V 中将相应的位置资源释放. **free** $s.e$ 将关联到方案 s 的特定项已执行完成的车工序移除,并从 h_T 中释放相应的车资源,但 h_V 中的位置资源保持不变.此外,注意到在操作语义的条件中,对两种资源堆的使用做了严格的限定约束,以确保格局转换均恰好集中于命令实际访问的资源.具体地,当一个车 t^a 被声明存在于 h_T 中时,所有由 t^a 指向的位置资源都需要出现在 h_V 中.并且,所有方案中出现的作业,也都需要存在于 h_T 中.上述设定符合分离逻辑的局部推理的思想,从而提高了分析和验证方案正确性的效率.

最后,给出产生资源调度冲突的格局转换.首先,对于方案命令 $s:=$ **asgn** (te^*) 和 **att** (s,te^*) ,若涉及的车资源不在 h_T 的定义域中,则会引起冲突;对于 **comp** s ,若方案 s 还未完全实施,则会引起冲突.车命令的情况比较复杂,其同时涉及车资源堆和位置资源堆.粗略地说,倘若没有预先激活相应的车资源和位置资源,则相关命令的执行均会引起潜在的资源调度冲突.形式化地,设 γ 为格局.记 $\gamma \rightsquigarrow \gamma'$ 表示从格局 γ 到另一个格局 γ' 的一步转移.记 $\gamma \rightsquigarrow^* \gamma'$ 表示存

在一个从 γ 到 γ' 的有限步转移.基于此,可以给出产生冲突格局 *stuck* 的 $\mathbf{ML}_{\mathbf{ASS}}$ 命令执行规则,如表 6 中所示.

表 6 产生冲突格局的 $\mathbf{ML}_{\mathbf{ASS}}$ 命令执行规则

方案规划:

$$\frac{[\![te_i]\!]\sigma \notin \mathrm{dom}(h_T)(1\le i\le n)}{\langle s:=\mathbf{asgn}(te_1,...,te_n),\sigma\rangle \rightsquigarrow stuck}$$

车工序附加:

$$\frac{(s_S(s)=(tt_1^a,...,tt_m^a),tt_j^a\notin \mathrm{dom}(h_T)(1\le j\le m))\text{或}\ ([\![te_i]\!]\sigma\notin\mathrm{dom}(h_T)(1\le i\le n))}{\langle \mathbf{att}\,(s,te_1,...,te_n),\sigma\rangle \rightsquigarrow stuck}$$

方案完成:

$$\frac{[\![s]\!]\sigma=(tt_1^a,...,tt_n^a)}{\langle \mathbf{comp}\ s,\sigma\rangle \rightsquigarrow stuck}$$

车工序增添:

$$\frac{([\![t^a]\!]\sigma\notin\mathrm{dom}(h_T))\text{或}\ (h_T([\![t^a]\!]\sigma)=(loc_1,...,loc_n),loc_i\notin\mathrm{dom}(h_V)(1\le i\le n))}{\langle \mathbf{add}\,(t^a,e),\sigma\rangle \rightsquigarrow stuck}$$

作业时长查询:

$$\frac{([\![te]\!]\sigma\notin\mathrm{dom}(h_T))\text{或}\ (h_T([\![te]\!]\sigma)=(loc_1,...,loc_n),loc_i\notin\mathrm{dom}(h_V)(1\le i\le n))}{\langle x:=\{te.e\},\sigma\rangle \rightsquigarrow stuck}$$

车资源更新:

$$\frac{([\![s]\!]\sigma=(tt_1^a,...,tt_n^a),tt_j^a\notin\mathrm{dom}(h_T)(1\le j\le n))\text{或}\ ([\![te]\!]\sigma\notin\mathrm{dom}(h_T))}{\langle s.e:=te,\sigma\rangle \rightsquigarrow stuck}$$

首项作业执行:

$$\frac{(h_T(tt_i^a)=(loc_1,...,loc_k),loc_1,...,loc_k\notin\mathrm{dom}(h_V))\text{或}\ ([e]\sigma=i,tt_i^a\notin_i^{SEQ}[\![s]\!]\sigma)\text{或}\ (tt^a\in^{SEQ}[\![s]\!]\sigma,tt^a\notin\mathrm{dom}(h_T))}{\langle \mathbf{1stexe}\ s.e,\sigma\rangle \rightsquigarrow stuck}$$

车资源移除:

$$\frac{([\![s.e]\!]\sigma=tt_k^a,s_S(s)=(tt_1^a,...,tt_k^a,...,tt_n^a),\ tt_1^a,...,tt_n^a\notin\mathrm{dom}(h_T))\text{或}\ (h_T(tt_k^a)\neq\mathbf{null})\text{或}(\text{对任意}s',tt_k^a\in^{SEQ}s_S(s'))}{\langle \mathbf{free}\ s.e,\sigma\rangle \rightsquigarrow stuck}$$

在上述定义的基础上,将安全格局定义为

$$\text{“}\langle C,\sigma\rangle \text{ 是安全的,”当}\langle C,\sigma\rangle \rightsquigarrow^* stuck \text{ 不可能发生.} \tag{16}$$

换言之,称 $\langle C,\sigma\rangle$ 是安全的,如果对于所有使得 $\langle C,\sigma\rangle \rightsquigarrow^* \gamma$ 的格局 γ ,格局 γ 不是特殊格局 *stuck* .

由此可见,本文的操作语义通过设置各种与资源堆相关的条件限制来要求空资源指向和未分配的车与位置资源均不能被间接引用,以此来保证计算的安全性性质,从而确保任何满足操作语义条件的命令执行都不会引发资源使用冲突.

综上所述,相比于现有的分离逻辑的存储-堆模型建模语言,本文的 $\mathbf{ML}_{\mathbf{ASS}}$ 主要有以下三个方面的区别.1)不同于分离逻辑语言致力于描述存储单元的突变操作, $\mathbf{ML}_{\mathbf{ASS}}$ 关注弹药保障作业的执行行为.2) $\mathbf{ML}_{\mathbf{ASS}}$ 的语义能够反映航母资源约束和保障流程规范.3) $\mathbf{ML}_{\mathbf{ASS}}$ 的车命令能够描述车和位置资源的多机、多任务共享.因此, $\mathbf{ML}_{\mathbf{ASS}}$ 具有足够的表达能力以描述弹药保障作业规划方案的实施过程,语义严格可用,可作为方案正确性验证的基础.

3 基于 Coq 的作业规划方案形式化验证

以第 2 节中的 $\mathbf{ML}_{\mathbf{ASS}}$ 为基础,本文在 Coq 中实现一个弹药保障作业规划方案正确性的证明辅助工具,并对前期工作[28]中所提出的作业规划方案进行交互式验证.

3.1 构建语法

类型推导是 Coq 证明系统的重要基础,为避免与字符串类型产生混淆,首先创建出用于声明变量名的 id 类型.

源码 1 声明变量名的编码

```
1. Inductive id : Type :=
2. | Id         : string -> id.
```

在 Coq 中将涉及到的值、位置资源和车资源均设置为自然数.并且,约定位置资源和车资源取为正整数,保留字集合 AToms 中的元素 **fin** 和 **null** 取为零和 nil 空序列.

$\mathbf{ML}_{\mathbf{ASS}}$ 的语法定义基于归纳定义,选择使用 Inductive 关键字对其进行形式化.在 Coq 中创建五个类型(算术)位置表达式 aexp、布尔表达式 bexp、车表达式 texp、方案表达式 sexp 和命令 command.具体定义如下.

源码 2 (算术)位置表达式的编码

```
1. Inductive aexp: Type :=
2. | ANum : nat -> aexp
3. | AId : id -> aexp
4. | APlus : aexp -> aexp -> aexp
5. | AMult : aexp -> aexp -> aexp
6. | AMinus : aexp -> aexp -> aexp.
```

类型 aexp 含有五个构造子:ANum、AId、APlus、AMult 和 AMinus,分别代表整数、变量、加法、乘法和减法,每个构造子均为一个函数类型,其中 nat 是 Coq 内置的自然数类型,“->”表示函数类型,如“A->B”是一个函数类型,将类型 A 的一个对象作为参数,返回类型 B 中的一个对象.此外,“->”在没有歧义的情况下可以省略,“A->B->C”表示参数类型为 A 且返回一个参数类型为 B->C 的函数类型,也可表示读入两个类型分别为 A 和 B 的参数,返回类型 C 的函数类型.在该归纳定义中,ANum 表示如果参数 a 是自然数类型,则 ANum a 是 aexp 类型;AId 表示如果 a 是 id 类型,则 AId a 是 aexp 类型;APlus 表示如果参数 a1 和 a2 都是 aexp 类型,则 APlus a1 a2 是 aexp 类型;AMult 表示如果参数 a1 和 a2 都是 aexp 类型,则 AMult a1 a2 是 aexp 类型;AMinus 表示如果参数 a1 和 a2 都是 aexp 类型,则 AMinus a1 a2 是 aexp 类型.

完成五个类型的归纳定义后,通过引入隐式转换 Coercion 和一些记法 Notation 以提升代码的可读性和易写性.通过使用 Coq 提供的 Coercion 声明为语法元素制定类型的隐式转换.例如,ANum 的转换声明在需要一个 aexp 时直接使用自然数,该自然数会被隐式地用 ANum 包装.通过使用 Coq 提供的 Notation 指令为语法元素制定符号编码,其中 at level 为他们定义优先级,但需避开 Coq 语言本身的保留符号,如“:=”,“;”等,且这些符号在具体的“记法作用域”中声明,以避免与其他符号相同的解释相冲突.例如,用 x ::=asgn y 代替 CSasgn x y,使其更贴合建模语言.具体如下.

源码 3 部分语法元素制定符号的编码

```
1. Notation "x + y" := (APlus x y) (at level 50, left associativity) : language_scope.
2. Notation "x - y" := (AMinus x y) (at level 50, left associativity) : language_scope.
3. Notation "x * y" := (AMult x y) (at level 40, left associativity) : language_scope.
```

```
Notation "x '::=asgn' y" := (CSasgn x y) (at level 80) : language_scope.
Notation "'att' ( x , y )" := (CSattach x y) (at level 80) : language_scope.
Notation "'comp' x" := (CScomp x) (at level 80) : language_scope.
Notation "x '::=plan' y" := (CTplan x y) (at level 80) : language_scope.
Notation "x '::=add' y" := (CTadd x y) (at level 80) : language_scope.
```

3.2 构建语义

在 Coq 中实现表 3 中所定义的储存和资源堆,具体如下.

源码 4 储存和资源堆的编码

```
Definition storeV := id -> nat.
Definition storeT := id -> nat.
Definition storeS := id -> list nat.
Definition heapV := nat -> option nat.
Definition heapT := nat -> option (list (option nat)).
```

由于冲突格局 *stuck* 与堆有关.基于此,运用 Coq 内置的可选类型,定义映射 heapV 和 heapT 的值域,以呼应堆是部分函数.该类型含有两个构造子,Some n 来表示正常的求值结果,None 来表示冲突的结果.对于 Some n 可使用函数 ov2 提取 n 进行使用,而对于冲突结果 None,可以按照使用场景进行不同的处理,这使得 option 成为一个可以包含冲突结果的类型.

基于此,采用(storeS x)的形式表示变量 x 在映射 storeS 中的求值,类似于 $s_S(x)$,这种表示方法也适用于其他映射.考虑到命令的执行会导致系统状态的更新,还定义了更新映射的函数,以映射 storeS 的更新函数(x !ss-> v ; stS)为例,它将作为变量 id 的 x 映射到值 v,并把这个映射关系增添到已有的映射 stS 中,那么对于变量 x 在函数更新后的映射结果,即((x !ss-> v ; stS)x)的求值为 v.其余四个映射的更新操作与之类似,分别以!sv->、!st->、!hv->和!ht->对应表示.

综上,根据式(4),在 Coq 中将系统状态 state 编码如下.

源码 5 系统状态的编码

```
Definition state : Type := (storeV * storeT * storeS * heapV * heapT).
```

接下来,在 Coq 中对 $\mathbf{ML}_{\mathbf{ASS}}$ 的语义进行编码.首先,编码 $\mathbf{ML}_{\mathbf{ASS}}$ 的表达式的求值函数. $\mathbf{ML}_{\mathbf{ASS}}$ 表达式的指称语义定义基于递归定义,选择使用 Fixpoint 关键字对其进行形式化.在 Coq 中创建 aeval、beval 和 teval 作为求值函数,分别用于描述 aexp、bexp 和 texp 类型在 Coq 中的语义.并且,求值函数接受映射作为额外的参数.具体如下.

源码 6 (算术)位置表达式的求值函数的编码

```
Fixpoint aeval (stoV: storeV) (a:aexp) : nat :=
match a with
| ANum n => n
| AId name => stoV name
| APlus a1 a2 => (aeval stoV a1) + (aeval stoV a2)
| ...
end.
```

该函数名称为 aeval,依赖于储存 storeV,函数返回自然数类型.而函数内部的构造包含一个模式匹配结构,其变

元就是递归函数的变元,即根据项 a 的结构进行匹配,并通过递归调用完成相应的求值.APlus 语义中的(aeval stoV a1)调用一种类型的映射求出表达式 a1 的值.

源码 7 布尔表达式的求值函数的编码

```
1. Fixpoint beval stoV (b:bexp) : option bool :=
2. match b with
3. | ...
4. | BEq a1 a2 => Some (beq_nat (aeval stoV a1) (aeval stoV a2))
5. | BLe a1 a2 => Some (leb (aeval stoV a1) (aeval stoV a2))
6. end.
```

其中函数 beq_nat 和函数 leb 是分别用于比较两个自然数是否满足“等于”和“小于等于”关系,其返回值是 bool 类型.

源码 8 车表达式的求值函数的编码

```
1. Fixpoint teval (stoV:storeV) (stoT:storeT) (stoS:storeS) (t:texp) : option nat :=
2. match t with
3. | ...
4. | TId name => Some (stoT name)
5. | TAddr sname a => findt (stoS sname) (aeval stoV a)
6. end.
```

其中函数 findt 返回方案 s 中相关联的车资源序列 te 中第 i 个车,而 te 和 i 分别是(stoS sname)和(aeval stoV a)的求值结果.teval 同时引入三种类型的映射 storeV、storeT 和 storeS,且该函数的求值结果为 option nat,其原因是如果 a 的求值结果大于方案相关联车资源序列的长度,会因为车资源未分配而导致空指针的调用,所以需要对这种情况所产生的冲突结果进行标记.

下面给出 $\mathbf{ML}_{\mathbf{ASS}}$ 命令的操作语义的编码.在 Coq 中创建 ceval 实现命令的语义,并在其中依据操作语义制定了 command 对系统状态的更新规则,即(ceval c st0 (St st’))表示为:系统初始状态为 st0 时,执行命令 c 并结束后,使系统状态更新为(St st’).注意到将终止状态设定为 ext_state 类型,与 option 类型的思想类似,该类型主要用于区分正常的终止状态(St st’)和冲突状态 *stuck* .

基于此,将 ceval 定义成一种“关系”而非一个“函数”,需要用户为某个程序求值成某种结束状态“构造证明”,而非只是交给 Coq 的计算机制去做.用命题类型 Prop 而非用 Type 定义它,可以理解为推导规则,“->”之前的若干命题都被视作规则中的前提.同时使用记法 st =[c]=> st'作为(ceval c st0 (St st’))的简记.为表述简单起见,下面给出部分命令的 Coq 编码与说明.

源码 9-1 方案规划命令的操作语义的编码

```
1. Inductive ceval: command -> state -> ext_state -> Prop :=
2. | ...
3. | E_Sasgn : forall stoV stoT stoS hV hT s t tloc,
4.               teval stoV stoT stoS t = Some tloc ->
5.                ceval (CSasgn s t) (stoV,stoT,stoS,hV,hT)
6.                       (St (stoV,stoT,(s !ss-> [tloc]; stoS),hV,hT))
```

该规则对应 $s:=\mathbf{asgn}(te^*)$ 的操作语义.若车表达式 t 对应带弹药类型的车资源为 tloc,那么执行该命令会在 stoS 中将方案变量 s 关联到车资源 tloc.

源码 9-2 车工序附加命令的操作语义的编码

```
1. | E_Sattach : forall stoV stoT stoS hV hT s t sss tloc,
2.                     teval stoV stoT stoS t = Some tloc ->
3.                     sss = stoS s ->
4.                     (forall l, in_list_list sss l = true -> exists k, hT l = k) ->
5.                     ceval (CSattach s t) (stoV,stoT,stoS,hV,hT)
6.                            (St (stoV,stoT,(s !ss-> (sss ++ [tloc]); stoS),hV,hT))
```

该规则对应 $\mathbf{att}(s,te^*)$ 的操作语义,其中符号"++"表示连接两个序列.根据当前的系统状态,求得车变量 t 所对应的车资源值 tloc,以及方案 s 关联的车资源序列 sss,判断车资源序列 sss 中的任意一个车资源 l 均在车资源堆 hT 的定义域中(判断函数 in_list_list),执行该命令会在 stoS 中将新增添的车资源 tloc 追加到 sss 尾部;注意如果 sss 为空,那么系统会直接新建以 tloc 为头节点的车资源序列,并关联到方案 s .

源码 9-3 方案完成命令的操作语义的编码

```
1. | E_Scomp : forall stoV stoT stoS hV hT s,
2.               nil = stoS s ->
3.               ceval (CScomp s) (stoV,stoT,stoS,hV,hT)
4.                      (St (stoV,stoT,(s !ss-> [fin];stoS),hV,hT))
```

该规则对应 $\mathbf{comp}\ s$ 的操作语义.若方案 s 关联的车资源序列是否为空,执行该命令会在 stoS 中将方案 s 设定为 fin,表明"被清空"的方案已实施完成.

源码 9-4 车工序规划命令的操作语义的编码

```
1. | E_Tplan : forall stoV stoT stoS hV hT t e n tloc loc,
2.             aeval stoV e = n ->
3.             hV loc = None ->
4.             hT tloc = None ->
5.             ceval (CTplan t e) (stoV,stoT,stoS,hV,hT)
6.                   (St (stoV, (t !st-> tloc; stoT), stoS,
7.                     (loc !hv-> n;hV), (tloc !ht-> [v2o loc];hT)))
```

该规则对应 $t^a:=\mathbf{plan}(\overline{e})$ 的操作语义.若位置 e 对应的作业时长值为 n,且该位置 loc 以及车资源 tloc 均未被分配,执行该命令会在 stoT 中添加车变量 t 其值为 tloc,在 hT 中将位置 loc 关联到新申请的车资源 tloc 中,并在 hV 中将位置 loc 中存入其对应的作业时长.

源码 9-5 车工序增添命令的操作语义的编码

```
1. | E_Tadd : forall stoV stoT stoS hV hT t e n loc tloc llist,
2.    (teval stoV stoT stoS t) = Some tloc ->
3.    hT tloc = Some llist ->
4.    (forall l, in_list llist l = true -> exists k, hV (o2v l) = k) ->
5.    aeval stoV e = n ->
6.    hV loc = None ->
```

```
7.     ceval (CTadd t e) (stoV,stoT,stoS,hV,hT)
8.     (St (stoV, stoT, stoS, (loc !hv-> n;hV), (tloc !ht-> llist ++ [(v2o loc)];hT)))
```

该规则对应 **add** (t^a,e) 的操作语义.根据当前的系统状态,求得车关联的位置序列 llist,且要求对于任意一个位置都属于 hV 的定义域;然后对位置表达式求值,求得新增位置所需的作业时长,且在该位置 loc 未被分配,执行该命令会在 hT 中将新增位置 loc 追加到 llist 尾部,并将该位置所需的作业时长注入到 hV.

源码 9-6 首项作业执行命令的操作语义的编码

```
1. | E_Texe : forall stoV stoT stoS hV hT lt t llist list h s sss,
2.                teval stoV stoT stoS t = Some lt ->
3.                hT lt = Some llist ->
4.                (forall l, in_list llist l = true -> exists k, hV (o2v l) = k) ->
5.                sss = stoS s ->
6.                (forall l, in_list_list sss l = true -> exists k, hT l = k) ->
7.                get_first (get_content llist) = Some h ->
8.                list_o_remove_one llist = list ->
9.                ceval
10.               (CTexe t) (stoV,stoT,stoS,hV,hT)
11.               (St (stoV,stoT,stoS,
12.               (hV_remove hV h),(hT_update hT lt list)))
```

该规则对应 **1stexe** *s.e* 的操作语义,其中函数 in_list 表示判断可选列表中是否有某个可选数,返回值为 bool 类型,若含有返回 true,否则返回 false;函数 get_first 表示返回序列中第一个元素;函数 get_content 表示将 list (option nat) 类型转换成 list nat 类型;函数 list_o_remove_one 表示返回序列 list (option nat)除去第一个元素以外的部分(即表尾);函数 hV_remove 表示将 hV 堆中某个特定的位置移出.首先判断方案 *s* 关联的车资源序列和该车关联的位置序列都在相应的定义域中,然后根据弹药保障的次序性,执行该车资源的首项保障作业,即将 hT 中现存车资源序列的首项移除,并将 hV 中相应的位置资源释放.

源码 9-7 车资源移除命令的操作语义的编码

```
1. | E_Tfree : forall stoV stoT stoS hV hT lt t llist s sss,
2.     sss = stoS s ->
3.     (forall l, in_list_list sss l = true -> exists k, hT l = k) ->
4.     teval stoV stoT stoS t = Some lt ->
5.     hT lt = Some llist ->
6.     nil = llist ->
7.     ceval (CTfree t) (stoV,stoT,stoS,hV,hT)
8.     (St (stoV,stoT,(s !ss-> rev(sS_remove_wei (rev sss) lt)++
9.     sS_remove_wei sss lt;stoS),hV,(hT_remove hT lt)))
```

该规则对应 **free** *s.e* 的操作语义,其中函数 rev 用于反转列表,函数 sS_remove_wei 和函数 hT_remove 均用于储存和堆中元素的移除.首先判断该方案 *s* 所关联的车资源序列 sss 中的任意一个车资源 l 都在车资源堆 hT 的定义域中,然后判断将要移除的车资源所关联的位置序列为空序列,执行该命令会将该车资源从 stoS 和 hT 中移除.

3.3 编码资源堆的性质

为方便处理验证过程中的格局转换,在 Coq 标准库的基础上对映射及其性质进行定义并证明.以堆 heapV 为例,具体的编码如下.例如定义了 hV_update_eq 引理表示更新堆 heapV 中某个键的映射,查寻该键即可得到更新后的值;定义了 hV_update_neq 引理表示更新堆 heapV 中某个键的映射,不影响其它键的映射;定义了 hV_update_shadow 引理表示连续两次更新堆 heapV 中同一键的映射,该键所关联的值仅与将第二次 update 应用于映射所得到的值表现一致等引理.对其余堆栈定义的引理与之类似.

源码 10 位置资源堆的性质的编码

```
1. Lemma hV_update_eq : forall heapV x v, (x !hv-> v ; heapV) x = Some v.
2. Lemma hV_update_neq :forall hV x1 x2 v, x1 <> x2 ->(x2 !hv-> v ; hV) x1 = hV x1.
3. Lemma hV_update_shadow : forall heapV x v1 v2, (x !hv-> v2 ; x !hv-> v1 ; heapV) =
4.    (x !hv-> v2; heapV).
5. Lemma hV_remove_neq : forall hV x1 x2 v, x1 <> x2 -> hV_remove (x2 !hv-> v;hV) x1 =
6.    (x2 !hv-> v; (hV_remove hV x1)).
7. Lemma hV_remove_work : forall hV x v, not_in_domV x hV -> hV_remove (x !hv-> v;hV) x = hV.
```

下面给出 hV_update_shadow 引理的证明过程,通过使用 functional_extensionality 策略对证明目标进行函数拓展,使用 unfold 策略对 hV_update 函数进行展开,以及利用 destruct 分类讨论待证目标中 beq_nat x x0 是否为真等 Coq 中的证明技巧来证明待证目标.

源码 11 hV_update_shadow 引理的证明

```
1. Lemma hV_update_shadow : forall heapV x v1 v2,
2.   (x !hv-> v2 ; x !hv-> v1 ; heapV) = (x !hv-> v2; heapV).
3. Proof.
4.   intros. apply functional_extensionality.
5.   intros. unfold hV_update.
6.   destruct (beq_nat x x0) eqn:H.
7.   trivial. trivial.
8. Qed.
```

3.4 方案验证实例

3.4.1 验证实例——方案部署

(1) 描述

表 7 方案部署算法

算法 1 方案部署	
1:	**function** DEPLOYMENT()
2:	$t_1^{A1} := \textbf{plan}(x_1, x_2, x_3)$;
3:	$t_1^{A2} := \textbf{plan}(y_1, y_2, y_3, y_4, y_5)$;
4:	$t_1^{A1r} := \textbf{plan}(x_4, x_5)$;
5:	$s_1 := \textbf{asgn}(t_1^{A1}, t_1^{A2}, t_1^{A1r})$;
6:	$x := \{s.2.3\}$;
7:	**end function**

如算法 1 中所示,本方案实例是部署一个方案并查询已部署方案中的特定位置上的作业时长.具体地,此算法首先规划一个由两种类型弹药转运车工序所构成的方案 s_1 ,依次规划 A1 类型的车 t_1^{A1} 承担的第一阶段任务,依照转运流程次序分配一个(隐式的)位置序列,且各位置所需的作业时长所构成的序列为 (x_1,x_2,x_3) ;规划 A2 类型的车 t_1^{A2} 全部五个阶段的转运任务;规划 A1 类型的车的第二阶段任务,记作 $t_1^{A1\prime}$;然后部署一个方案 s_1 ,该方案被关联到由 $(t_1^{A1},t_1^{A2},t_1^{A1\prime})$ 表示的一组互不相同的(已规划的)车资源所构成的时序性序列,其中的每个序列项(即一个车资源)代表方案的一道工序;最后查询已完成部署的方案 s_1 的第 2 道工序中第 3 个阶段位置上的作业时长.

(2) 证明过程

交互式的定理证明通过平台 CoqIDE 进行实现,左边部分为脚本缓冲区以显示由用户输入的 Coq 代码,右边部分为目标窗口在证明过程中给出当前的证明状态,其中横线上方为当前环境的前提,下方为待证目标.下面结合代码说明 DEPLOYMENTs1_Correct 在证明辅助工具中的证明步骤.

a) 待证事实描述

首先使用关键字 Lemma 描述 DEPLOYMENTs1_Correct 引理,表示该引理的名称为 DEPLOYMENTs1_Correct,待证目标的含义为对于任意互不相等且不为零的车变量、车资源和位置,若系统的初始状态为空 empty_st,则执行 DEPLOYMENTs1 并终止后,描述系统五元组计算状态的各个分量表述如下.

(x!sv->1002; null_sV): Stores_V 中变量 x 暂存值为 1002,即已完成部署的方案 s_1 的第 2 道工序中第 3 个阶段位置所需作业时长为 10 分 02 秒.

(t1_A1' !st-> tt1_A1'; t1_A2 !st-> tt1_A2; t1_A1 !st-> tt1_A1; null_sT): Stores_T 中车变量 t1_A1'、t1_A2 和 t1_A1 暂存值分别为 tt1_A1'、tt1_A2 和 tt1_A1.

(s1 !ss-> [tt1_A1;tt1_A2;tt1_A1']; null_sS,): Stores_S 中方案变量 s1 关联的互不相同的(已规划的)车资源所构成的时序性序列[tt1_A1;tt1_A2;tt1_A1'].

(loc_51_A1!hv-> 0517 ;…; emp_heapV): Heaps_V 中位置 loc_51_A1 所需作业时长为 05 分 17 秒,其余位置同理.

(tt1_A1'!ht-> [v2o loc_41_A1;v2o loc_51_A1];…; emp_heapT): Heaps_T 中车资源 tt1_A1'所关联的位置序列为[v2o loc_41_A1;v2o loc_51_A1],其中函数 v2o 表示将 nat 类型转变为 option nat 类型,其余车资源同理.为方便说明和阅读,下文以 final_st 简记系统待证的终止状态.

在该命令后,使用一个可选命令 Proof,使得该会话的脚本更易阅读,此时 Coq 进入证明模式,结合当前上下文和待证目标,系统会创建一个原始待证目标,在显示目标时,用一条水平线将上下文和待证目标上下分割开,用户可应用策略不断地对该目标进行拆解,进而完成自底向上的证明.

b) 引入假设,展开证明

使用 unfold 策略对函数 neq_tt3、neq_loc10 等按照定义展开后,使用 intros 策略引入目标中的前提 H~H62,表示各项资源间互不相等且不为零,被加入水平线上方,用户仅需解释如何运用这些前提去构造结论的证明.

c) 证明方案部署

首先,通过引入参数 loc_11_A1,运用 eapply E_Tplan with (loc := loc_11_A1)策略进入车工序规划命令的证明,使用 eapply 指令,调用顺序命令自动以类似?st 的形式,暂时引入所需的变量,用户可以在后续证明中将其改为确切值,由于系统初始状态显然满足车 t1_A1 规划命令的相应语义条件,因此可直接触发语义规则进行状态转换.同时在证明过程中,还考虑到运用 Coq 内置的自动化验证方式来缩短验证过程;其次在其产生的每个子目标中执行 reflexivity 策略完成证明,同时使用映射引理如 hT_update_eq、sS_update_shadow 等实现对系统状态的化简;然后依次执行车 t1_A2 和 t1_A1'规划;最后利用 eapply E_Sasgn with (tloc := tt1_A1)、eapply E_Sattach with (tloc := tt1_A1')等策略完成方案规划,即该方案被关联到一组互不相同的带弹药类型的车工序序列且车资源在车资源堆中已定义完全,并获得一个中间状态,注意到 Stores_V 仍为空.

方案部署的证明过程以及此时上下文和待证命题如图 3 所示.

Coq

图 3　方案部署证明过程

d) 证明作业时长查询

使用 eapply E_Tlookup 策略进入作业时长查询命令的证明,由于上述中间状态满足所涉及资源堆的存在性,因而保证了相应语义规则可以被触发,首先利用 Coq 内置的自动验化验证方式确认查询的车资源;然后使用 apply SafeinHt58z 策略作用于待证目标,完成对车资源 tt1_A2 所关联的位置序列均在位置资源堆定义域的证明;最后使用 reflexivity 策略、hV_update_neq 和 hV_update_eq 引理确认查询的保障位置 loc_32_A2 以及位置上的作业时长.

e) 完成证明

至此,Coq 提示“无更多子目标”,即方案部署与特定位置作业时长查询证明已完成,如图 4 所示.

f) 退出证明

最后,使用 Qed 命令声明证明已完成,并使 Coq 退出证明模式.

Coq 程序代码	当前证明状态
Proof. …(*接图 3 代码*) eapply E_Tlookup. simpl; auto. rewrite hT_update_neq. rewrite hT_update_eq. reflexivity. auto. intros l L0. apply SafeinHt58z;auto. simpl. reflexivity. reflexivity. rewrite hV_update_neq. rewrite hV_update_neq. rewrite hV_update_neq. rewrite hV_update_neq. rewrite hV_update_eq. reflexivity. auto. auto. auto. auto.	No more subgoals.

图 4　方案部署证明结束

3.4.2　验证实例——车工序执行

(1) 描述

表 8　车工序执行算法

算法 2　车工序执行	
1:	**function** PROCESSEXE(s,i)
2:	$j := \#s.i$;
3:	**while** $1 \le j$ **do**
4:	**1stexe** $s.i$;
5:	$j := j - 1$;
6:	**end while**
7:	**end function**

车工序是方案流程调度的基本单元,并以保障位置序列为构成要素,描述了弹药保障系统运行过程中的基本业务流程.如算法 2 中所示,该实例的输入是规划方案中的特定项车工序,其输出为该项车工序被置空的方案,实现了对

车工序下任意长度的位置资源序列的释放操作.具体地,此算法首先读取方案 s 的第 i 道车工序所关联到的转运目标位置的数目,然后进入 while 循环从首部逐一执行各个位置的保障作业任务,即将与该车工序相关联的所有位置资源从堆 $\text{Heaps}_{l'}$ 中移除.

(2) 证明过程

以 3.4.1 小节中方案 s_1 所关联的第 1 项车工序为例,证明车工序执行算法.使用 eapply E_Seq 策略,进入 PROCESSEXE 函数,使用 eapply E_Tlength 策略求得车承担转运目标位置的数目 *len* ,使用 eapply E_Ass 策略将转运目标位置数目赋值给变量 j .

a) 进入第一次循环

使用 eapply E_WhileLoop 策略,进入车工序执行算法的第一次循环,使用 reflexivity 策略确认此时循环判定条件 $1 \leq j$ 为真.

b) 执行循环体

使用 eapply E_Texe 策略进入首项作业执行命令的证明,首先确认执行的车资源,利用 hT_update_eq 和 SafeinHt3 引理确认车资源所关联的保障位置序列以及证明该保障位置序列在位置资源堆中的存在性;其次确认方案所关联的车资源序列,利用 Inllist 引理和存在量化 exists 证明该方案所关联的车资源序列满足在车资源堆中的存在性;然后利用 hV_remove_neq 和 hV_remove_work 引理将该车资源所关联的首项位置从位置资源堆中移除,并利用 hT_update_shadow 引理对该车资源所关联的保障位置序列进行更新;最后更新循环变量的值,即执行 $j-1$ 操作,并确认已移除保障位置不在位置资源堆中.

车工序执行中第一次循环的证明过程以及此时上下文和待证命题如图 5 所示.

Coq

图 5 车工序 t_1^{A1} 第一次循环的证明过程

c) 进入第二次和第三次循环

第二次和第三次循环证明过程类似,此处不再赘述.

d) 循环终止

使用 eapply E_WhileEnd 策略证明循环可终止,即确认循环条件为假,如图 6 所示.

Coq

图 6 循环终止证明过程

3.4.3 验证实例——方案实施

(1) 描述

表 9 方案实施算法

算法 3 方案实施	
1:	**function** IMPLEMENTATION()
2:	$t_1^{A1} := \mathbf{plan}(x_1, x_2, x_3)$;
3:	$s_1 := \mathbf{asgn}(t_1^{A1})$;
4:	PROCESSEXE (s_1,1);
5:	**free** s_1.l;
6:	$t_1^{A2} := \mathbf{plan}(y_1, y_2, y_3, y_4, y_5)$;
7:	**att**(s_1, t_1^{A2});
8:	PROCESSEXE (s_1,1);
9:	**free** s_1.l;
10:	$t_1^{A1r} := \mathbf{plan}(x_4, x_5)$;
11:	**att**(s_1, t_1^{A1r});
12:	PROCESSEXE (s_1,1);
13:	**free** s_1.l;
14:	**comp** s_1;
15:	**end function**

如算法 3 中所述,本方案实例是对 3.4.1 中方案的实施.具体地,此算法首先规划一个由两种类型弹药转运车工序所构成的方案 s_1 ,其中一类车转运依照方案时序要求拆分成了两个工序阶段,而且各车工序均依照保障阶段次序规划了各保障位置的弹药转运时间;并在规划完一个车工序后,执行该车工序,将与该车资源相关联的所有位置资源从堆 Heaps_V 中移除,并将该车资源从堆 Heaps_T 中移除;当方案 s_1 的内容为空,可触发方案完成命令.

(2) 证明过程

在 3.4.1 小节的证明过程中已经展示了该证明辅助工具对方案部署的证明过程,本小节将对有关方案部署的证明进行一定程度的省略,重点对方案实施的证明步骤进行详细说明.

a) 待证事实描述

首先对于任意互不相等且不为零的车变量、车资源和位置,若系统的初始状态为空 empty_st,则执行 IMPLEMENTATIONs1 并终止后,堆 Heaps_T 和 Heaps_V 仍为空, Stores_S 中 s_1 的值变为 **fin** ,表示方案 s_1 完成,与 s_1 相关联的车资源均变为可寻但已释放资源.

b) 证明车工序执行

证明过程与 3.4.2 小节类似,此处不再赘述.

c) 证明车资源移除

在证明过程中,通过引入参数 s1 明确指定了值的变量,运用 eapply E_Tfree with (s := s1)策略进入对车资源移除命令的证明,首先利用 sS_update_eq、Inllist 等引理确认方案所关联的车资源序列以及证明车资源序列满足在车资源堆中的存在性;然后利用 hT_update_eq 和 reflexivity 引理确认移除的车资源所关联的保障位置序列为空;最后利用 hT_remove_work 和 beq_refl 引理从车资源堆中释放相应的车资源,并利用 sS_update_shadow 引理更新该方案所关联的车资源序列.

车资源移除的证明过程以及此时上下文和待证命题如图 7 所示.

Coq

图 7　车资源移除的证明过程

d) 证明方案完成以及完成方案实施的证明

使用 eapply E_Scomp 策略进入方案完成命令的证明,利用 sS_update_eq 引理确认方案内容为空和 sS_update_shadow 引理更新该方案所关联的车资源序列,即将方案设定为 fin.同时利用 reflexivity 策略完成对使用 eapply E_Tfree 策略产生的子目标的证明,即证明车资源不在空 Heaps_T 堆中.此时显示 No more subgoals.完成作业规划方案执行的证明.

e) 退出证明

使用 Qed 命令声明证明已完成,并使 Coq 退出证明模式.

3.4.4　验证实例——多任务协同调度

综合上述实例,可以形式化图 2 所给出的调度示例.本例考虑如下情形,当方案 s_1 部署并实施后,此时系统状态为式(17),即当前状态中保障位置 loc_{52} 未释放,转运车 $tt_1^{\text{A1}\prime}$ 未移除,系统部署一套“新”方案 s_2 ,即将方案 s_1 此时的状态作为方案 s_2 执行的初始状态.

$$([s_S \mid s_1 : tt_1^{\text{A1}\prime}],[s_T \mid t_1^{\text{A1}\prime} : tt_1^{\text{A1}\prime} \mid t_1^{\text{A2}} : tt_1^{\text{A2}} \mid t_1^{\text{A1}} : tt_1^{\text{A1}}],s_V,[h_T \mid tt_1^{\text{A1}\prime} : \textbf{null}],[h_V \mid loc_{52} : 1000]). \tag{17}$$

验证结果表明,与方案 s_2 相关联的车 tt_2^{A1} 仍可正常部署与实施,这是因为即使转运车 $tt_1^{\text{A1}\prime}$ 未在方案 s_1 完成后被释放,由于方案 s_2 争取到的车资源 tt_2^{A1} 是一个“新”车,且与其相关联的位置均已被释放或是“新”位置.从而未引起“使用未释放资源”的错误.证明完成如图 8 所示.本实例展示了本方法能够分析和验证弹药保障系统中的多任务协同调度.

Coq

图 8　方案 s_2 中转运车 tt_2^{A1} 部署与实施证明结束

3.4.5　验证实例——多任务间共享资源常占情形

依然以图 2 为例,本例考虑多任务间的潜在资源常占情形.当一套方案实施后,但其所关联的部分资源在弹药保障系统中未释放,此时部署另一套方案就可能存在常占资源无法被再次使用,具体如下.

(1) IMPLEMENTATIONs2_Abt1 的描述

部署与方案 s_2 相关联的转运车 $tt_1^{\text{A1}\prime}$,但其无法被“重部署”,因其在方案 s_1 中未移除,继而无法满足车工序规划命令操作语义中 $tt^a \in \mathcal{T} - \text{dom}(h_T)$ 条件,即待规划转运车需未在 Heaps_T 堆中,从而在 Coq 的证明过程中产生下列矛盾公式:(tt1_A1' !ht-> []; emp_heapT) tt1_A1' = None.该式中,等号左边的含义为转运车 tt1_A1'在 emp_heapT 映射中

的求值为[]显然不等于 None,导致证明中断,如图 9 所示.

Coq

图 9　方案 s_2 中转运车 $tt_1^{A1\prime}$ 部署失败

(2) IMPLEMENTATIONs2_Abt2 的描述

部署与方案 s_2 相关联的转运车 tt_1^{A2},虽其可被部署,保障位置 loc_{43} 为已释放位置可被使用,但因该车工序规划中,保障位置 loc_{52} 为未释放资源,继而无法满足车工序规划命令操作语义中 $loc_1,...,loc_n \in \mathrm{Loc} - \mathrm{dom}(h_{V'})$ 条件,即待规划转运车所关联的保障位置需未在 $\mathrm{Heaps}_{V'}$ 堆中,从而在 Coq 的证明过程中产生下列矛盾公式:(loc_52 !hv-> 1000; emp_heapV) loc_52 = None.该式中,等号左边的含义为保障位置 loc_52 在 emp_heapV 映射中的求值为 Some 1000 显然不等于 None,导致证明中断,如图 10 所示.

综上,本实例展示了本方法能够发现弹药保障系统动态申请与释放共享资源行为下潜在的具有隐匿性、累积性的资源占用漏洞,从而避免资源利用率下降.

Coq

图 10　方案 s_2 中转运车 tt_1^{A2} 部署失败

总的来说,在 Coq 中实现弹药保障作业规划方案的正确性证明辅助工具,包含 95 条弹药保障系统行为定义以及 71 条配套的引理,并对 5 种典型方案进行交互式证明.具体的代码统计如表 10 所示.

表 10　Coq 代码行统计

描述	Coq 代码行数
弹药保障系统模型	2033
验证	1324
总数	3357

4　结论

为了保证航母舰载机弹药保障作业规划方案的正确性,本文提出了一个符合弹药保障系统特征的双层资源堆形式化模型,并且以此为基础构造了一套可用于描述作业执行行为的建模语言,并给出了其操作语义.通过将该弹药保障模型形式化地实现于 Coq 证明辅助工具中,本文交互式验证了 5 个作业规划方案验证实例,共涉及 3357 行 Coq

代码,包括 95 条定义和 71 条引理,完整的代码源文件可见 https://github.com/jinlu07/AssVerifi.git.验证结果表明,所提出的建模语言具有足够的表达能力,所实现的 Coq 工具能够支持正确性证明的机械化校验并简化了证明过程.

结合目前工作存在的不足,本文对后续工作做出以下展望:(1)本文实现的是常态下弹药保障系统的确定性行为建模,但在现实中,航海环境错综复杂、战场态势瞬息万变[29],使得该情形下系统表现出不确定性行为,需进一步对其行为进行概率随机建模与定量描述.(2)本文所关注的弹药保障系统是航母航空指挥和保障系统(简称航保系统)的一个二级系统[30],对于整个航保系统的建模、分析和验证,以及两个系统的耦合影响机理,也是未来的重点研究工作之一.(3)通过编制更多的自动化脚本来提升证明效率,并以优化编码的方式来提升代码的可读性.

References:

[1] Li YF, Wu QS, Xu ML, Lv P, Jiang XH, Zhu RJ, Zhou B. Real-time scheduling for carrier-borne aircraft support operations: a reinforcement learning approach. Scientia Sinica Inform, 2021,51(2):247–262 (in Chinese with English abstract).

[2] Zhang L, Gao P, Du ZP, Liu HY, Wang J. Research on aircraft carrier damage evaluation criteria based on quantitative analysis of aviation support capability. Ship Science and Technology, 2022,44(11):185–189 (in Chinese with English abstract).

[3] Xiao HB. Research on the safety of ammunition stowage and handling aboard aircraft carriers. Aerodynamic Missile Journal., 2019(4):85–88 (in Chinese with English abstract).

[4] Tian DH, He JM, Qi J, Sun HX. Research on the modeling and simulation of optimal dynamic aerial ammunition scheduling and transportation. Journal of Northwestern Polytechnical University, 2018,36(6):1236–1242 (in Chinese with English abstract).

[5] Liu A, Liu K. Advances in carrier-based aircraft deck operation scheduling. Systems Engineering Theory & Practice, 2017,37(1):49–60 (in Chinese with English abstract).

[6] Tao JQ, Su XC, Han W, Li YF. Study of aircraft carrier ammunition scheduling optimization based on EDA algorithm. Journal of Ordnance Equipment Engineering, 2022,43(5):125–131 (in Chinese with English abstract).

[7] Wang J, Zhan NJ, Feng XY, Liu ZM. Overview of formal methods. Ruan Jian Xue Bao/Journal of Software, 2019,30(1):33–61 (in Chinese with English abstract).

[8] Wang SL, Zhan BH, Sheng HH, Wu H, Yi SC, Wang LT, Jin XY, Xue B, Li JH, Xiang SQ, Xiang Z, Mao BF. Survey on requirements classification and formalization of trustworthy systems. Ruan Jian Xue Bao/Journal of Software, 2022,33(7):2367–2410 (in Chinese with English abstract).

[9] Zhan NJ, Wang J. Challenges and trends for specification, analysis, and verification of complex systems. Science and Technology Foresight, 2023,2(1):7–22 (in Chinese with English abstract).

[10] Reynolds J C. Separation Logic: A logic for shared mutable data structures. In: Proc. of the 17th IEEE Symp. on Logic in Computer Science. Copenhagen: Institute of Electrical and Electronics Engineers, 2002. 55–74.

[11] Pym D, Spring JM, O'Hearn PW. Why separation logic works. Philosophy & Technology, 2019,32(3):483–516.

[12] O'Hearn PW. Separation logic. Communications of the ACM, 2019,62(2):86–95.

[13] Chen H, Ziegler D, Chajed T, Chlipala A, Kaashoek M F, Zeldovich N. Using Crash Hoare logic for certifying the FSCQ file system. In: Proc. of the 25th Symposium on Operating Systems Principles. New York: Association for Computing Machinery, 2015. 18–37.

[14] Xu FW, Fu M, Feng XF, Zhang XR, Zhang H, Li ZH. A practical verification framework for preemptive OS kernel. In: Proc. of the Int'l Conf. on Computer Aided Verification. Cham: Springer International Publishing, 2016. 57–59.

[15] Amani S, Bégel M, Bortin M, Staples M. Towards verifying Ethereum smart contract bytecode in Isabelle/HOL. In: Proc. of the 7th ACM SIGPLAN Int'l Conf. on Certified Programs and Proofs. New York: Association for Computing Machinery, 2018. 66–77.

[16] Le QL, Raad A, Villard J, Berdine J, Dreyer D, O'Hearn PW. Finding real bugs in big programs with incorrectness logic. Proceedings of the ACM on Programming Languages, 2022,6(OOPSLA1):1–27.

[17] Distefano D, Fähndrich M, Logozzo F, O'Hearn P W. Scaling static analyses at Facebook. Communications of the ACM, 2019,62(8):62–70.

[18] Cao QX, Beringer L, Gruetter S, Dodds J, Appel AW. VST-Floyd: A separation logic tool to verify correctness of C programs. Journal of Automated Reasoning, 2018,61(1):367–422.

[19] Feng XY, Shao Z, Dong Y, Guo Y. Certifying low-level programs with hardware interrupts and preemptive threads. ACM SIGPLAN Notices, 2008,43(6):170–182.

[20] Zhan BH. AUTO2, a saturation-based heuristic prover for higher-order logic. In: Proc. of the Int'l Conf. on Interactive Theorem Proving. Cham: Springer International Publishing, 2016. 441–456.

[21] Zhang LP, Zhao YW, Wang BY, Li YX, Feng XX. Formal verification of the virtual memory subsystem in L4. Ruan Jian Xue Bao/Journal of Software, 2023,34(8):3527–3548 (in Chinese with English abstract).

[22] Wang XB, Kou MS, Li CY, Zhao L. Implementation of theorem prover for PPTL with indexed expressions. Ruan Jian Xue Bao/Journal of Software, 2022,33(6):2172–2188 (in Chinese with English abstract).

[23] Li YN, Deng YX, Liu J. Formal modeling and verification of Paxos based on Coq. Ruan Jian Xue Bao/Journal of Software, 2020,31(8):2362–2374 (in Chinese with English abstract).

[24] Su WY, Gao C, Gu XC, Wu ZL. COMPSPEN: Separation logic solver for integrated reasoning about shape properties and data constraints. Ruan Jian Xue Bao/Journal of Software, 2023,34(5):2181–2195 (in Chinese with English abstract).

[25] Zhang BW, Jin Z, Wang HP, Cao YZ. A tool for verifying cloud block storage based on separation logic. Ruan Jian Xue Bao/Journal of Software, 2022,33(6):2264–2287 (in Chinese with English abstract).

[26] Liu Y, Han W, Su X, Cui R. Optimization of fixed aviation support resource station configuration for aircraft carrier based on aircraft dispatch mission scheduling. Chinese Journal of Aeronautics, 2023,36(2):127–138.

[27] Jin Z, Zhang BW, Cao TY, Cao YZ, Wang HP. Reasoning about block-based cloud storage systems via separation logic. Theoretical Computer Science, 2022,936:43–76.

[28] Zhang SH, Liu S, Li YF, Jin Z, Jin YY, Wang SC, Zhao JB, Xu ML. Optimization algorithm for ammunition support operation scheduling of carrier-borne aircraft. Acta Aeronautica et Astronautica Sinica, 2023,44(20):228485 (in Chinese with English abstract).

[29] Li GF. Design analysis about ammunition boarding safely. Ship Science And Technology, 2022,44(2):166–169 (in Chinese with English abstract).

[30] Li YF, Gao L, Hao HJ, Jin YY, Wang K, Xu ML. Human-machine collaborative decision-making for carrier aircraft support operations. Scientia Sinica Inform, 2023,53(12):2493–2510 (in Chinese with English abstract).

附中文参考文献:

[1] 李亚飞,吴庆顺,徐明亮,吕培,姜晓恒,朱睿杰,周兵.基于强化学习的舰载机保障作业实时调度方法.中国科学:信息科学,2021,51(2):247–262.

[2] 张磊,高鹏,杜志鹏,刘海燕,王健.基于航空保障能力定量分析的航母毁伤评估判据研究.舰船科学技术,2022,44(11):185–189.

[3] 肖虎斌.航母弹药贮运安全作业研究.飞航导弹,2019(4):85–88.

[4] 田德红,何建敏,齐洁,孙海信.航空弹药动态调运决策优化建模与仿真研究.西北工业大学学报,2018,36(6):1236–1242.

[5] 刘翱,刘克.舰载机保障作业调度问题研究进展.系统工程理论与实践,2017,37(1):49–60.

[6] 陶俊权,苏析超,韩维,李亚飞.基于 EDA 算法的航母弹药调度优化研究.兵器装备工程学报,2022,43(5):125–131.

[7] 王戟,詹乃军,冯新宇,刘志明.形式化方法概貌.软件学报,2019,30(1):33–61.

[8] 王淑灵,詹博华,盛欢欢,吴昊,易士程,王令泰,金翔宇,薛白,李静辉,向霜晴,向展,毛碧飞.可信系统性质的分类和形式化研究综述.软件学报,2022,33(7):2367–2410.

[9] 詹乃军,王戟.复杂系统规约、分析与验证发展现状与展望.前瞻科技,2023,2(1):7–22.

[21] 章乐平,赵永望,王布阳,李悦欣,冯潇潇.L4 虚拟内存子系统的形式化验证.软件学报,2022,34(8):3527–3548.

[22] 王小兵,寇蒙莎,李春奕,赵亮.支持索引式的 PPTL 定理证明器的实现.软件学报,2022,33(6):2172–2188.

[23] 李亚男,邓玉欣,刘静.基于 Coq 的 Paxos 形式化建模与验证.软件学报,2020,31(8):2362–2374.

[24] 苏婉昀,高冲,古新才,吴志林.COMPSPEN:对形状性质与数据约束进行融合推理的分离逻辑求解器.软件学报,2023,34(5):2181–2195.

[25] 张博闻,金钊,王捍贫,曹永知.一种基于分离逻辑的块云存储系统验证工具.软件学报,2022,33(6):2264–2287.

[28] 张少辉,刘舜,李亚飞,金钊,靳远远,王少参,赵建波,徐明亮.航空母舰舰载机弹药保障作业调度优化算法.航空学报,2023,44(20):228485.

[29] 李冠峰.弹药安全上舰设计分析.舰船科学技术,2022,44(2):166–169.

[30] 李亚飞,高磊,蒿宏杰,靳远远,王可,徐明亮.舰载机保障作业人机协同决策方法.中国科学:信息科学,2023,53(12):2493–2510.