Contents

1 研究内容 1.2：面向集成测试的需求约束序列智能构建方法

航天嵌入式软件集成测试需求涉及跨模块业务流程，包含多步骤协作、状态转换与时序控制等复杂逻辑。现有方法在并发执行、异常分支、状态同步等场景下，易出现序列不完整和逻辑不一致。为此，本项目提出流程感知的约束序列智能构建机制，通过语义分析与时序推理，建立流程节点模型和执行约束框架，生成结构化约束序列，为测试脚本自动生成提供坚实基础。

1.1 基于状态机建模的业务流程约束序列构建方法

针对业务流程表达多样、层次复杂、逻辑隐含等问题，提出状态机与有向图结合的流程建模方案。利用自然语言处理技术抽取业务实体、操作动作和状态转换条件，结合领域知识补全隐含流程信息，构建顺序执行、并发控制、条件分支和异常处理的综合状态转换模型，建立节点依赖关系和时序约束，形成可验证的约束序列结构。

1.2 语义关键词驱动的约束序列优化与质量保障机制

针对序列语义信息不足与逻辑一致性缺失，建立语义关键词标注体系，涵盖操作、对象、状态和关系等核心维度。通过实体识别、语义角色标注和依赖关系抽取，为序列节点赋予精确语义标签。设计序列优化算法，采用模型检测验证状态可达性，利用约束求解器检测逻辑冲突，构建融合语义覆盖度、逻辑一致性与执行可达性的质量评估体系，提升约束序列的匹配效率与执行精度。

2 研究内容 1.2-技术路线

针对航天嵌入式软件集成测试需求中业务流程复杂、约束分散、序列构建难的问题，本课题拟采用” 流程语义解析-状态机建模-序列智能构建” 的技术路线：首先，通过深度语义分析识别业务流程中的约束要素与逻辑关系，构建多层次的约束分类体系；其次，采用扩展有限状态机建模复杂业务流程，支持并发控制和异常处理；最后建立多维语义标注机制，生成支持智能匹配的高质量约束序列。具体研究方案如下：

表 1: 面向集成测试的需求约束序列智能构建技术路线

步骤	描述
集成测试需求输入	接收集成测试需求，作为输入数据
流程语义解析	对需求进行语义解析，提取流程中的约束要素
约束要素提取	提取约束要素，包括操作、对象、状态和关系等
状态机建模	基于扩展有限状态机对业务流程进行建模
约束序列构建	构建约束序列，支持并发控制、异常处理和路径优化
多维语义标注	对约束序列进行多维语义标注，提升语义信息的完整性
序列质量评估与优化	评估序列质量，优化逻辑一致性、覆盖度和执行效率
高质量约束序列输出	输出经过优化的高质量约束序列，支持测试脚本自动生成

2.1 流程语义解析与约束要素提取技术

针对集成测试需求中业务流程表达多样、约束类型混杂，导致的约束识别不准确、分类不完整的问题，本课题拟采用” 语义分层解析-约束类型识别-关系依赖建模” 的技术路线：

2.1.1 多层次流程语义解析与约束边界识别方法

实现复杂业务流程的精确语义解析，需要解决两个核心问题：1）集成测试需求中的业务流程描述往往跨越多个功能模块，存在隐含的执行依赖和状态转换，传统的线性文本分析方法难以捕捉跨句子的流程逻辑；2）约束信息在需求语句中表达分散，同一约束可能在多个句子中被部分描述，缺乏统一的约束边界识别机制。

针对问题 1），构建基于依存句法分析的跨句语义关联模型，Relation 由句法分析和时序连接词共同决定；引入句间语义角色标注机制，通过 Agent-Action-Object 三元组抽取跨句的流程要素。

针对问题 2），设计约束边界识别算法：采用 BIO 标注体系标记约束起始边界，结合 Constraint-Trigger 词典识别约束触发词，通过分散片段合并函数生成完整约束。

2.1.2 约束类型智能分类与验证机制

为保障约束分类的准确性与完整性，需要解决以下核心问题：1）不同类型约束在语言表达上存在交叉和重叠，单一分类器难以准确区分顺序约束、并发约束、异常约束等复合类型；2）约束分类结果缺乏逻辑一致性验证，可能出现相互矛盾的约束类型标注。

针对问题 1），构建层次化多标签分类器，分为粗粒度和细粒度分类，并引入注意力机制增强分类器对关键语言模式的感知能力。

针对问题 2），建立约束逻辑一致性检验机制，定义约束兼容性矩阵，通过约束求解器验证可满足性，自动检测和修复约束冲突。

2.2 状态机驱动的约束序列构建技术

针对现有序列构建方法在处理并发控制、异常分支时结构表达能力不足的问题，本课题拟采用” 状态机建模-序列图构建-可执行性验证” 的技术路线：

2.2.1 扩展有限状态机的业务流程建模方法

实现复杂业务流程的状态机建模，需要解决以下核心问题：1）传统有限状态机难以表达并发执行和资源共享场景，在航天嵌入式软件中多个任务可能同时访问共享资源，需要支持并发状态和同步机制；2）业务流程中的异常处理逻辑复杂，包括超时、错误恢复、回滚等多种异常控制路径，状态机需要支持异常状态转换。

针对问题 1），构建并发状态机模型，包含并发状态集合和状态转换保护条件；引入资源访问控制机制处理资源竞争。

针对问题 2），设计异常状态处理机制：为每个正常状态配备对应的异常状态，通过异常转换函数定义异常处理逻辑。

2.2.2 约束序列图自动构建与路径优化方法

实现高效的约束序列构建，需要解决以下核心问题：1）从状态机模型到可执行序列的转换过程中，需要处理状态爆炸和路径组合爆炸问题，特别是在存在循环和并发的情况下；2）生成的约束序列需要满足时序约束和资源约束，但传统的序列构建方法缺乏全局优化机制。

针对问题 1），采用状态空间裁剪策略，通过状态可达性分析减少不可达状态，利用路径合并方法合并语义等价路径，并优先处理复杂度较低的路径。

针对问题 2），建立全局序列优化模型：

$$\text{OptimalSequence} = \arg\min(\text{Length}(\text{seq}) + \lambda \times \text{Conflict}(\text{seq})) \tag{1}$$

其中：

- Length 表示序列长度，
- Conflict 表示约束冲突程度，
- λ 为权重参数；采用遗传算法求解最优序列构建方案。

2.3 多维语义标注与序列质量保证技术

针对约束序列语义标注不充分、匹配精度不高的问题，本课题拟采用” 层次化标注-质量评估-持续优化” 的技术路线：

2.3.1 多维语义关键词自动抽取与标注方法

实现高质量的语义标注，需要解决两个核心问题：1）约束序列中的语义信息多层次、多维度，包括操作语义、对象语义、状态语义等，需要建立统一的语义标注框架；2）领域专业术语丰富且不断演化，人工标注成本高、更新滞后，需要自动化的术语发现和标注机制。

针对问题 1），构建语义关键词标注架构，分别包含通用操作词汇、航天嵌入式专业术语和具体场景描述；采用多头注意力机制融合不同层次的语义信息。

针对问题 2），设计术语自动发现算法，利用 TF-IDF 和 PMI 计算词汇重要性和共现强度，通过 Word2Vec 聚类发现语义相似术语，并建立术语库增量更新机制。

2.3.2 序列质量评估与自动优化机制

为保障约束序列的实用性，需要解决以下核心问题：1）约束序列的质量评估缺乏统一标准，现有方法多关注语法正确性，忽略语义完整性和执行效率；2）序列质量缺陷的自动修复能力不足，依赖人工干预，影响系统的实用性和可扩展性。

针对问题 1），建立多维质量评估模型：

$$\text{Quality}_{\text{score}} = \alpha \times \text{Completeness} + \beta \times \text{Consistency} + \gamma \times \text{Efficiency} \tag{2}$$

其中：

- Completeness 衡量序列覆盖度，
- Consistency 衡量逻辑一致性，
- Efficiency 衡量执行效率；采用 AHP 层次分析法确定权重参数。

针对问题 2），构建缺陷自动修复机制，通过 DefectClassifier 自动识别缺陷类型，建立修复规则库，并采用强化学习算法优化修复策略，逐步提升修复成功率。

3 应用示例

本节通过两个具体的航天嵌入式软件集成测试场景，展示所提出的” 流程语义解析-状态机建模-序列智能构建” 技术路线在实际应用中的有效性。示例涵盖了飞行任务调度和卫星数据采集两个典型场景，分别体现了并发资源管理和状态监控异常处理的复杂约束序列构建需求。

3.1 示例 1：飞行任务调度约束序列构建

技术路线对应关系：本示例主要体现了多层次流程语义解析、扩展有限状态机建模以及多维语义标注技术的综合应用，重点展示了并发控制和动态调度约束的处理能力。

3.1.1 需求描述

飞行任务调度需要根据任务优先级动态分配资源，确保高优先级任务优先执行，同时在资源不足时触发任务排队机制，并在任务完成后释放资源。

3.1.2 技术路线应用过程

1. 流程语义解析阶段：

- 应用多层次流程语义解析技术，识别” 优先级判断”、” 资源分配”、” 任务排队” 等关键约束要素
- 通过约束边界识别算法，提取” 任务提交 → 优先级判断 → 资源分配 → 任务执行 → 资源释放” 的完整流程链
- 应用约束类型智能分类，将需求分类为并发约束、顺序约束和动态调度约束

2. 状态机建模阶段：

- 构建扩展有限状态机，支持并发状态：” 等待队列”、” 优先级排序”、” 资源分配中”、” 任务执行中”
- 引入资源访问控制机制，处理多任务并发访问共享资源的竞争问题
- 设计状态转换保护条件，确保高优先级任务的优先调度逻辑

3. 序列智能构建阶段：

- 应用路径优化算法，生成最优的任务调度序列
- 进行多维语义标注，为每个节点标注操作语义（监控、判断、分配、释放）
- 实施质量评估与优化，确保序列的逻辑一致性和执行效率

3.1.3 约束序列构建分析过程

识别为基于优先级调度的任务管理流程，包含资源监控、优先级判断、任务分配和资源释放等逻辑。通过流程分析确定调度规则、资源分配策略和任务完成后的资源回收机制。生成包含动态调度和资源管理的复合约束序列。

3.1.4 构建的标准化约束序列

- **序列标识：**FLIGHT_TASK_SCHEDULING_SEQUENCE
- **序列类型：**优先级调度约束序列

状态监控约束：

- 监控对象：任务队列和资源池
- 监控条件：任务提交事件发生
- 状态更新：任务队列中新增任务，资源池状态更新

执行节点约束：

1. **节点 1 - 任务队列监控：**
   - 语义标签：[监控, 队列, 任务, 提交]
   - 执行约束：持续监控任务队列中的任务提交情况
   - 触发条件：任务提交事件发生
   - 状态约束：维护任务队列的实时状态
   - 输出数据：当前任务队列状态
2. **节点 2 - 优先级判断：**
   - 语义标签：[判断, 优先级, 排序, 调度]
   - 执行约束：根据任务优先级对任务队列进行排序
   - 前置约束：任务队列中存在未分配任务
   - 判断逻辑：高优先级任务优先分配资源
   - 输出数据：排序后的任务队列
3. **节点 3 - 资源分配：**
   - 语义标签：[分配, 资源, 调度, 执行]
   - 执行约束：为高优先级任务分配可用资源
   - 前置约束：资源池中存在可用资源
   - 状态约束：资源池状态更新，任务状态变更为执行中
   - 输出数据：任务执行状态和资源分配结果
4. **节点 4 - 任务完成监控：**
   - 语义标签：[监控, 完成, 任务, 释放]
   - 执行约束：监控任务完成情况并释放资源
   - 前置约束：任务状态为执行中
   - 状态约束：任务状态变更为完成，资源池状态更新
   - 输出数据：任务完成状态和资源释放结果

执行顺序约束：节点 1 持续监控，节点 2 优先级判断，节点 3 资源分配，节点 4 任务完成监控

动态调度约束：任务队列动态排序，资源分配优先级实时调整

状态同步约束：任务完成后释放资源并更新任务队列状态

3.2 示例 2：卫星数据采集流程约束序列构建

技术路线对应关系：本示例重点体现了状态机驱动的序列构建技术和序列质量保证机制，特别展示了异常处理约束和状态同步约束的建模与验证能力。

3.2.1 需求描述

卫星数据采集流程需要依次完成传感器初始化、数据采集、数据存储和数据传输四个步骤。要求在数据采集过程中支持实时监控采集状态，并在传输失败时重新尝试。

3.2.2 技术路线应用过程

1. 流程语义解析阶段：

- 应用约束要素提取技术，识别” 初始化”、” 采集”、” 存储”、” 传输” 等核心操作要素
- 通过跨句语义关联分析，建立” 初始化 $\rightarrow$ 采集 $\rightarrow$ 存储 $\rightarrow$ 传输” 的时序依赖关系
- 识别异常处理约束：” 传输失败 $\rightarrow$ 重试机制”

2. 状态机建模阶段：

- 构建状态转换模型：传感器准备、数据采集中、数据存储中、数据传输中、传输失败、任务完成
- 设计异常状态处理机制，为传输状态配备对应的异常状态和重试逻辑
- 建立状态同步约束，确保各阶段状态的正确转换和数据一致性

3. 序列智能构建阶段：

- 应用序列质量评估模型，确保采集流程的完整性和一致性
- 进行语义标注，标识实时监控、异常处理等关键语义特征
- 构建自动优化机制，处理传输失败的重试策略优化

3.2.3 约束序列构建分析过程

识别为基于状态监控的采集流程，包含初始化、实时监控、数据存储和异常处理等逻辑。通过流程分析确定采集状态监控机制、存储策略和传输失败的重试机制。生成包含实时监控和异常处理的复合约束序列。

3.2.4 构建的标准化约束序列

- **序列标识**：SATELLITE_DATA_COLLECTION_SEQUENCE
- **序列类型**：状态监控约束序列

状态监控约束：

- 监控对象：数据采集状态和传输状态
- 监控条件：数据采集和传输事件发生
- 状态更新：实时更新采集状态和传输状态

执行节点约束：

1. **节点 1 - 传感器初始化：**
   - 语义标签：[初始化, 传感器, 准备, 状态]
   - 执行约束：初始化传感器并准备采集环境
   - 触发条件：采集任务启动
   - 状态约束：传感器状态变更为已准备
   - 输出数据：传感器初始化状态
2. **节点 2 - 数据采集：**
   - 语义标签：[采集, 数据, 实时, 监控]
   - 执行约束：实时采集数据并监控采集状态
   - 前置约束：传感器状态为已准备
   - 状态约束：采集状态实时更新
   - 输出数据：采集数据和采集状态
3. **节点 3 - 数据存储：**
   - 语义标签：[存储, 数据, 持久化, 状态]
   - 执行约束：将采集的数据存储到本地存储设备
   - 前置约束：采集状态为完成
   - 状态约束：存储状态更新为已完成
   - 输出数据：存储状态和存储结果
4. **节点 4 - 数据传输：**
   - 语义标签：[传输, 数据, 网络, 重试]

- 执行约束：通过网络传输数据，并在失败时重试
- 前置约束：存储状态为已完成
- 状态约束：传输状态更新为成功或失败
- 输出数据：传输状态和传输结果

执行顺序约束：节点 1 初始化，节点 2 实时采集，节点 3 存储，节点 4 传输

异常处理约束：节点 4 传输失败时触发重试机制

状态同步约束：传输成功后更新采集任务状态为完成

3.2.5 技术路线验证效果

通过以上两个示例的应用，验证了所提出技术路线的有效性：

- **语义解析准确性**：能够准确识别并分类复杂的约束类型，包括并发约束、异常约束和状态同步约束
- **状态机建模完整性**：扩展有限状态机能够有效表达并发执行、异常处理等复杂业务逻辑
- **序列构建质量**：生成的约束序列具备良好的逻辑一致性、语义完整性和执行效率
- **实用性验证**：构建的约束序列能够直接支持测试脚本的自动生成，提升了集成测试的自动化水平