故障树领域本体及SWRL规则的构建方法研究

周 亮 黄志球 黄传林

(南京航空航天大学计算机科学与技术学院 南京 210016)

摘 要 故障树(Fault Tree,FT)被广泛应用于系统故障的快速定位,但其因缺乏精确的语义信息而存在重复构建问题。将本体引入到故障树领域中,并对如何构建故障树本体及相应的SWRL规则进行了研究:首先采用本体描述语言(Web Ontology Language,OWL)对故障树中的概念及概念之间的关系进行知识表示,构建了一个可共享、可重用、可扩展的故障树领域本体;然后将故障树中事件之间的逻辑关系转化成语义Web规则语言(Semantic Web Rule Language,SWRL);最后将构建的故障树领域本体和SWRL规则放入JESS推理机中进行推理,产生新的知识,用于系统故障的快速定位。实验证明,使用所提出的方法能在解决故障树重复构建问题的同时,不对系统故障的快速定位产生影响。

关键词 本体,故障树,SWRL,推理

中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.8.042

Construction Method for Fault Tree Domain Ontology Supporting SWRL Rules

ZHOU Liang HUANG Zhi-qiu HUANG Chuan-lin

(School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)

Abstract Fault tree (FT) has been widely applied in rapid location of system faults. However, lacking of accurate semantic information makes it hard to avoid the problem of duplicate construction. This paper introduced ontology into fault tree domain and studied how to construct a FT domain ontology and SWRL. First, it proposed a method of knowledge representation of FT with Web ontology language OWL and constructed a FT domain ontology which is shareable, reusable and extensible. Second, it transformed the logical relationship among events of FT into semantic Web rule language(SWRL). Finally, it put these SWRL rules and the FT ontology into an inference engine JESS. Then new knowledge is produced and it is exploited for the rapid location of system faults. The causes of events can be rapidly and efficiently located in FT. The experiment proves the correctness and effectiveness of proposed method. It can resolve the problem of duplicate construction of FT without any influence on the rapid location of system faults.

Keywords Ontology, Fault tree, SWRL, Inference

1 引言

故障树[1]是一种定性的逻辑关系因果图。它将系统最不希望发生的故障作为顶事件,把不能再分解的基本故障作为基本事件,把顶事件和基本事件之间的一切事件称为中间事件。用相应的符号代表这些事件,再用适当的逻辑门把这些事件连接成树状图,这种倒立的树状图就称作故障树。但由于构建故障树的知识来源多且分散,比如专家、书籍、网络等,这些知识往往是异构的,相互之间缺乏精确的语义信息,很容易造成故障树的重复构建问题。

本体是对概念以及概念之间关系的精确描述,在知识的重用和共享方面优势明显,近年来愈来愈受到人们的关注。其理论基础和相关技术也在不断地完善与发展,在许多领域已经得到了广泛的应用。目前,国外建立的本体非常多,已经实现的本体主要有:Cyc[2]、WordNet[3]、UMLS[4]等。Cyc项目是由Douglas Lenat在1984年设立的,由Cycorp公司开发并维护,其目标是建立一个庞大的人类常识知识库,使人工智能的应用能够以类似人类推理的方式工作。WordNet是由Princeton大学的心理学家、语言学家和计算机工程师联合设计的一种基于认知语言学的英语词典,它不光是把单词以字母顺序排列,而且按照单词的意义组成一个"单词的网络",并且支持自动的文本分析以及人工智能应用。UMLS又称为统一医学语言系统,是对生物医学科学领域内许多受控词表的一部纲目式汇编。UMLS提供的是一种位于这些词表之间的映射结构,目的是使这些不同的术语系统之间能够彼此转换。国内对本体的研究起步较晚,与国外相比,还存在一些差距,但本体也逐渐受到学者们的关注,成为一个研究热点。Ge等[5]对隐私领域进行了详细的分析,建立了隐私领域本

到稿日期:2014-08-03 返修日期:2014-10-21 本文受国家自然科学基金(61100034,61170043),中央高校基本科研业务费专项资金(CXZZ11_0218)资助。

周 亮(1983-),男,硕士生,主要研究方向为基于本体的软件工程、语义Web,E-mail:zhouliang588344@sohu.com;黄志球(1965-),男,博士生导师,CCF高级会员,主要研究方向为软件工程、软件度量;黄传林(1988-),男,硕士生,主要研究方向为软件安全、形式化方法。

体，将 Web 服务中用户的隐私偏好转化成 SWRL 规则，通过 JESS 推理引擎进行推理，得到满足用户隐私偏好的 Web 服务，能较好地保护用户的隐私信息。Huang 等[6]提出了使用可扩展访问控制标记语言 XACML 进行扩展的方法，引入 OWL 语言来表达 RBAC 中的角色约束，设计并构建了访问控制领域知识库，利用知识库领域本体的一致性检测来进行约束管理和冲突检测。Zhong 等[7]总结了若干几何学知识表示模式和获取模式，在此基础上构建了一个较为完善的几何学领域本体，为几何学专家系统等领域提供了智能基础。Chen[8]等人提出了一种重用现有领域知识库知识来构造新领域本体的方法，该方法利用领域知识模型以及领域本体相互之间存在的语义相关性，从语义匹配的角度探讨了构造新领域本体的可能性。

本文首先对故障树领域进行分析，提取出故障树中的主要构成要素，确定了构成要素的层次结构，并在此基础上，借助本体构建工具 Protégé 手工构建了一个故障树领域本体；然后详细介绍了如何根据故障树中事件之间的逻辑关系（用逻辑门来表示）来构建相应的 SWRL 规则；最后，以一个故障树为例，将故障树领域本体和相应的 SWRL 规则一起放入 JESS 推理机中进行推理，从而得到顶事件发生的原因（也就是系统发生故障的原因）。实验结果表明，本文提出的方法是正确和有效的。

本文第 2 节介绍本体相关理论；第 3 节构建了故障树领域本体和 SWRL 规则；第 4 节是实验分析；最后总结全文并指出进一步工作。

2 本体相关理论

2.1 本体的构建方法

本体（Ontology）的概念起源于哲学领域，近年来逐渐成为计算机及相关领域关注的一个研究热点。然而，到目前为止，本体还没有统一的定义。Gruber 将本体定义为“概念模型的明确的规范”，得到了许多同行的认可，并被广泛引用。从定义可以看出，本体是用来描述某个领域内的概念以及概念之间的关系，使得这些概念和概念间的关系在共享的范围内具有大家共同认可的、明确的、唯一的定义[9]。本体目前被广泛应用于语义 Web、人工智能、信息获取和电子商务等领域。

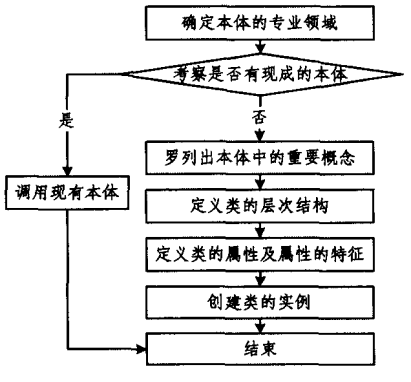

图 1 七步法流程图

目前，本体工程还不成熟，很多本体工程都使用自己的方法来开发各自的本体，比较有名的有骨架法、企业建模（TOVE）法、Methontology 法、七步法等。本文需要构建一个故障树领域本体，因此采取七步法。七步法[10]是由斯坦福大学开发的，主要用于领域本体的构建。该方法的 7 个步骤分别是：1）确定本体的专业领域和范畴；2）考察复用现有本体的可能性；3）列出本体中的重要术语；4）定义类和类的等级体系；5）定义类的属性；6）定义属性取值的类型、容许的取值、取值的个数和有关属性取值等其他特征；7）创建类的实例。七步法的流程图如图 1 所示。

2.2 SWRL 规则

本体属于数据层，用来描述领域资源，但很难表达类属性之间的关系。因此，需要采用规则语言对本体数据进行规则扩展，以扩充本体的隐含知识，使本体具有实际应用价值。SWRL[11]规则语言由 Daml. org 组织提出，于 2004 年 5 月 21 日正式成为 W3C 的成员提案之一。SWRL 是由 RuleML[12]演变而来，是以 OWL 子语言 OWL DL[13]、OWL Lite 和 RuleML 为基础的规则描述语言，能够方便地将推理规则和本体结合在一起，使本体表达的内容更为丰富，更具有实用性。SWRL 规则的基本形式是表示前提和结论的推导关系。前提和结论都可以包括单个或者多个基本命题，基本命题之间是逻辑与的关系。SWRL 框架中的 Atom 用于定义条件判断的限制式，而 SWRL 框架中的 lmp 用于定义规则，lmp 中的限制式由 Atom 提供，在本体中 SWRL 规则主要使用两个限制式：

（1）$C(x)$：x 可以是变量或本体的实例，C 是类，说明 x 是 C 类的一个实例，如 *Parent*(?x)，说明 Parent 是一个类。

（2）$P(x,y)$：x、y 可以是变量或本体的实例，P 是对象属性。如 *hasBrother* (?x,?y)，说明 y 是 x 的兄弟。

举例说明如下：

hasFather(?x,?y) ∧ *hasBrother*(?y,?z)→*hasUncle*(?x,?z)

在上述 SWRL 规则中，x、y、z 为 3 个不同的实例，*hasParent*、*hasBrother*、*hasUncle* 为 3 个不同的对象属性。该规则的前提是：如果 y 是 x 的父亲，并且 z 是 y 的兄弟，则结论是：z 是 x 的叔叔。

3 故障树本体的创建

3.1 故障树领域

根据七步法，需要考察复用现有本体的可能性。目前，并没有现成的故障树领域本体，需要自己构建。因此，本节首先对故障树领域进行分析，罗列出故障树中的重要概念。

故障树领域包含两个要素：事件（Event）和逻辑门（Logic_gates），逻辑门用于表达事件与事件之间的逻辑关系。本文根据故障树领域的特点并结合实际使用情况，把故障树分为故障树名称（Fault_tree_name）、事件（Event）、事件状态（Event_status）、事件的种类（Event_species）、逻辑门（Logic_gates）、逻辑门的种类（Logic_gate_species）6 个平行的基本类。本文采用斯坦福大学的本体构建工具 Protégé 来构建本体。Protégé 是本体研究者构建本体的首选工具，提供了构建本体的基本功能，其由于优秀的设计和众多的插件而成为目前主流的本体构建工具。故障树本体的类层次结构如图 2 所示。

定义了类后，定义类的属性。属性分为对象属性和数据

属性。本文构建的故障树本体，是用来判断系统发生故障的原因，不需要定义数据属性。Protégé 工具的 Properties 标签页用于创建类属性，其中 Object 标签页用于创建对象属性。对象属性主要包括 fault_tree_name_is、event_species_is、Status_is、condition_is、Logic_gate_species_is、output_event_is、input_event_1_is、input_event_2_is、input_event_3_is 等。部分对象属性的详细信息如表 1 所列。

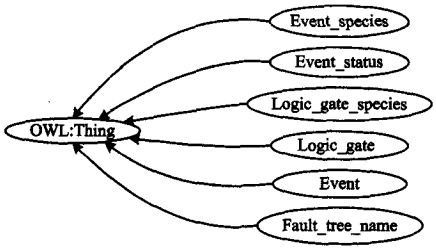

图 2　故障树本体类层次结构图

表 1　故障树本体的部分对象属性详细信息

属性名	定义域	值域	属性说明
fault_tree_name_is	Event、Logic_gate	Fault_tree_name	故障树名
event_species_is	Event	Event_species	事件的种类是
Status_is	Event	Event_status	事件的状态是
condition_is	Logic_gate	Event	禁门和优先与门中输出事件发生的条件是
Logic_gate_species_is	Logic_gate	Logic_gate_species	逻辑门的种类是
output_event_is	Logic_gate	Event	逻辑门的输出事件是
input_event_1_is	Logic_gate	Event	逻辑门的第一个输入事件是

最后，根据具体的故障树，选择 Protégé 工具的 Individuals标签页，就可以对本体创建实例并添加实例的属性值。通过上述步骤，故障树本体初步创建完成。本体构建完成后，还需要对其进行一致性验证，以确保本体中包含的所有知识不存在矛盾。本文采用 pellet 推理机对构建的故障树本体进行检查，结果是满足一致性要求的。经验表明，本体的创建不是一步到位的，在使用过程中，需要对本体不断优化和完善，以便更好地适应实际使用情况。

3.2　构建 SWRL 规则

故障树中，事件之间的逻辑关系是用逻辑门来表示的，逻辑门分为与门、或门、非门、优先与门、表决门、异或门和禁门 7 种。针对这 7 种逻辑门，本文分别给出了对应的 SWRL 规则，并做简单说明。

a)与门(And_gate)：与门的输入事件全部发生，输出事件才会发生。逻辑与门表示的是事件的交集。

假设逻辑门 G 为与门，A 是与门 G 的输出事件，B 和 C 是与门 G 的输入事件。如果输出事件 A 发生，由与门的定义可知：输入事件 B 和输入事件 C 一定会同时发生。

SWRL 规则如 Rule-1(规则 1)所示：

Rule-1：Logic_gate(?x) ∧ Logic_gate_species_is(?x,与门) ∧ output_event_is(?x,?a) ∧ input_event_1_is(?x,?y) ∧ input_event_2_is(?x,?z) ∧ Status_is(?a,发生)→Status_is(?y,发生) ∧ Status_is(?z,发生)

在规则 1 中：Logic_gate、Logic_gate_species_is、output_event_is、input_event_1_is、input_event_2_is、Status_is 来源于 3.1 节构建的故障树本体中的类和对象属性。

对规则 1 解析如下：Logic_gate(?x)表示 x 是 Logic_gate 类的一个实例；Logic_gate_species_is(?x,与门)表示 x 为与门；output_event_is(?x,?a)表示实例 x 的 output_event_is 为实例“a”；input_event_1_is(?x,?y)和 input_event_2_is(?x,?z)表示实例 x 的 input_event_1_is、input_event_2_is 分别为实例“y”和实例“z”；Status_is(?a,发生)表示实例 a 的 Status_is 为实例“发生”；Status_is(?y,发生)和 Status_is(?z,发生)表示实例 y 和 z 的 Status_is 都为实例“发生”。后文的 SWRL 规则与规则 1 类似，限于篇幅，不再做详细解析。

b)或门(Or_gate)：一个或者多个输入故障发生时，输出事件即可发生。逻辑或门表示的是事件的并集。

假设逻辑门 G 为或门，A 是或门 G 的输出事件，B 和 C 是或门 G 的输入事件。如果输出事件 A 发生，由或门的定义可知，输入事件会有 3 种可能情况：①B 事件发生，C 事件不发生；②B 事件不发生，C 事件发生；③B 事件和 C 事件都发生。

当逻辑门为或门，在不知道输入事件是否发生时，本文默认第一个输入事件发生，其余的输入事件不发生(根据事件发生的概率，按照从大到小的原则，依次为第一个输入事件、第二个输入事件等，这样可以使故障定位更加准确)，将推理结果和实际情况进行比较，如果二者不一致，则第二个输入事件发生，其余不发生。依此类推，来建立 SWRL 规则。异或门和表决门构建 SWRL 规则的思路与或门相同。

SWRL 规则如 Rule-2、Rule-3 所示：

Rule-2：Logic_gate(?x) ∧ Logic_gate_species_is(?x,或门) ∧ output_event_is(?x,?y) ∧ input_event_1_is(?x,?z) ∧ input_event_2_is(?x,?a) ∧ Status_is(?y,发生) ∧ Status_is(?a,不发生)→Status_is(?z,发生)

Rule-3：Logic_gate(?x) ∧ Logic_gate_species_is(?x,或门) ∧ output_event_is(?x,?y) ∧ input_event_1_is(?x,?z) ∧ input_event_2_is(?x,?a) ∧ Status_is(?y,发生) ∧ Status_is(?z,不发生)→Status_is(?a,发生)

c)非门(Not_gate)：非门的输出事件是将输入事件取反后得到的对立事件。

假设逻辑门 G 为非门，A 是非门 G 的输出事件，B 是非门 G 的输入事件。如果输出事件 A 发生，由非门的定义可知：输入事件 B 不发生。

SWRL 规则如 Rule-4 所示：

Rule-4：Logic_gate(?x) ∧ Logic_gate_species_is(?x,非门) ∧ output_event_is(?x,?y) ∧ input_event_1_is(?x,?z) ∧ Status_is(?y,发生)→Status_is(?z,不发生)

d)优先与门(Priority_and_gate)：全部输入故障按照特定顺序发生时，输出事件发生。(事件发生的顺序由逻辑门右边的条件事件规定)。在输出事件不变的前提下，可将优先与门用与门替换，并将右侧约束“优先与”顺序的条件事件作为一个新的基本事件输入。

假设逻辑门 G 为优先与门，A 是优先与门 G 的输出事件，B 和 C 是优先与门 G 的输入事件，D 为条件事件。如果输出事件 A 发生，由优先与门的定义可知：输入事件 B、输入

事件 C 和条件事件 D 同时发生。

SWRL 规则如 Rule-5 所示：

Rule-5：Logic_gate(?x) ∧ Logic_gate_species_is(?x,优先与门) ∧ output_event_is(?x,?y) ∧ input_event_1_is(?x,?z) ∧ input_event_2_is(?x,?a) ∧ condition_is(?x,?b) ∧ Status_is(?y,发生)→Status_is(?z,发生) ∧ Status_is(?a,发生) ∧ Status_is(?b,发生)

e)异或门(Xor_gate)：表示当两个输入事件有且仅有一个事件发生时，输出事件才发生。两个输入事件都发生或都不发生时，均不能导致输出事件的发生。

假设逻辑门 G 为异或门，A 是异或门 G 的输出事件，B 和 C 是异或门 G 的输入事件。如果输出事件 A 发生，由异或门的定义可知，输入事件会有 2 种可能情况：①B 事件发生，C 事件不发生；②B 事件不发生，C 事件发生。

SWRL 规则如 Rule-6、Rule-7 所示：

Rule-6：Logic_gate(?x) ∧ Logic_gate_species_is(?x,异或门) ∧ output_event_is(?x,?y) ∧ input_event_1_is(?x,?z) ∧ input_event_2_is(?x,?a) ∧ Status_is(?y,发生) ∧ Status_is(?a,不发生)→Status_is(?z,发生)

Rule-7：Logic_gate(?x) ∧ Logic_gate_species_is(?x,异或门) ∧ output_event_is(?x,?y) ∧ input_event_1_is(?x,?z) ∧ input_event_2_is(?x,?a) ∧ Status_is(?y,发生) ∧ Status_is(?z,不发生)→Status_is(?a,发生)

f)禁门(Exclusion_gate)：当逻辑门左侧(也可以在右侧)给定的“条件事件”满足时，输入事件的发生直接引起输出事件发生；若“条件事件”不满足，则输入事件不能引发输出事件的发生。

假设逻辑门 G 为禁门，A 是禁门 G 的输出事件，B 是禁门 G 的输入事件，C 为条件事件。如果输出事件 A 发生，由禁门的定义可知：输入事件 B 和条件事件 C 同时发生。

SWRL 规则如 Rule-8 所示：

Rule-8：Logic_gate(?x) ∧ Logic_gate_species_is(?x,禁门) ∧ output_event_is(?x,?y) ∧ input_event_1_is(?x,?z) ∧ condition_is(?x,?a) ∧ Status_is(?y,发生)→Status_is(?a,发生) ∧ Status_is(?z,发生)

g)表决门(Voting_gate)：该逻辑门表示当 n 个事件中有 r 个事件或 r 个以上事件发生时，输出事件发生，否则不发生。

假设逻辑门 G 为表决门，A 是表决门 G 的输出事件，B、C 和 D 是表决门 G 的输入事件，r 为 2。如果输出事件 A 发生，由表决门的定义可知，输入事件会有 4 种可能情况：①B 和 C 事件发生，D 事件不发生；②B 和 D 事件发生，C 事件不发生；③C 和 D 事件都发生，B 事件不发生；④B、C、D 3 个事件同时发生。

SWRL 规则如 Rule-9、Rule-10、Rule-11 所示：

Rule-9：Logic_gate(?x) ∧ Logic_gate_species_is(?x,表决门) ∧ output_event_is(?x,?y) ∧ input_event_1_is(?x,?z) ∧ input_event_2_is(?x,?a) ∧ input_event_3_is(?x,?b) ∧ Status_is(?y,发生) ∧ Status_is(?z,不发生)→Status_is(?a,发生) ∧ Status_is(?b,发生)

Rule-10：Logic_gate(?x) ∧ Logic_gate_species_is(?x,表决门) ∧ output_event_is(?x,?y) ∧ input_event_1_is(?x,?z) ∧ input_event_2_is(?x,?a) ∧ input_event_3_is(?x,?b) ∧ Status_is(?y,发生) ∧ Status_is(?a,不发生)→Status_is(?z,发生) ∧ Status_is(?b,发生)

Rule-11：Logic_gate(?x) ∧ Logic_gate_species_is(?x,表决门) ∧ output_event_is(?x,?y) ∧ input_event_1_is(?x,?z) ∧ input_event_2_is(?x,?a) ∧ input_event_3_is(?x,?b) ∧ Status_is(?y,发生) ∧ Status_is(?b,不发生)→Status_is(?z,发生) ∧ Status_is(?a,发生)

通过上述方法就能将故障树中事件之间的逻辑关系构建成相对应的 SWRL 规则。将构建的故障树本体和 SWRL 规则放入 JESS 推理机中进行推理，能得到本体中的隐含知识，从而实现对系统故障的快速定位。

4 案例分析

本文构建了一个有“与门”、“或门”、“非门”、“优先与门”、“异或门”、“禁门”和“表决门”7 种逻辑门的故障树，如图 3 所示。首先，可以假设在这个故障树中顶事件 T 发生，并且 $M4$ 事件经过确认，没有发生；然后，根据故障树分析法可知，T 事件发生，有两种可能：①$M1$、$M2$、$M3$、$M5$、$C1$、$C2$、$B1$、$B2$ 和 $B6$ 同时发生；②$M1$、$M2$、$M3$、$M6$、$C1$、$B1$ 和 $B2$ 同时发生，$B7$ 不发生。

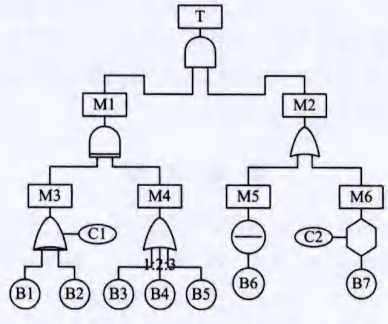

图 3 某一待分析的故障树

接着，将 3.1 节中构建的故障树领域本体和 3.2 节中构建的 11 条 SWRL 规则一起加载到 Protégé 工具中的 JESS 推理机中进行推理。在 3.2 节构建 SWRL 规则时提到，本文在遇到逻辑或门时，默认逻辑或门的第一个输入事件发生，即 B6 事件发生。实验结果如图 4 所示，与用故障树分析法得到的第一种可能相同。

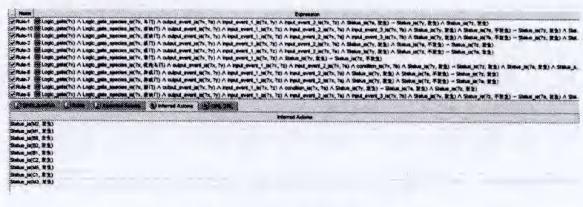

图 4 B6 事件发生的实验结果

最后，根据推理得到的实验结果，对系统进行故障检查。如果发现 B6 事件并没有发生，那么将检查结果反馈到本体中，再次通过 JESS 推理机进行推理，得到第二种可能。实验结果如图 5 所示。

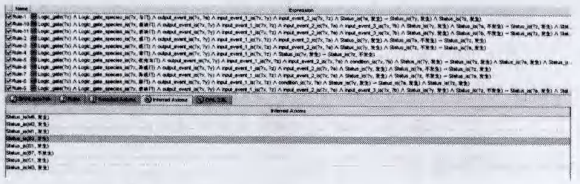

图 5　B6 事件不发生的实验结果

通过比较发现，利用本文提出的方法对故障树进行分析，实验结果与用故障树分析法得出的结果是一致的，说明本文提出的方法是可行的。

结束语　使用故障树分析法对故障树进行分析，能极大地提高硬件的可靠性和安全性，在系统安全工程中发挥着重要的作用。故障树分析法除了能对系统故障进行快速定位外，还能通过计算事件发生的概率，来找出系统的薄弱环节，从而提高系统的安全性。前者属于故障树分析法的定性分析，后者属于故障树分析法的定量分析。但故障树本身存在一些缺陷，例如重复构建问题。

本体是对概念以及概念之间关系的精确描述，在知识的重用和共享方面优势明显，能很好地解决故障树的重复构建问题。但是，目前还没有一个大规模、可共享、可重用的故障树本体。因此，建立一个良好的故障树本体具有重要的意义。本文根据“七步法”构建了一个故障树领域本体，在此基础上，对如何根据事件之间的逻辑关系来构建相应的 SWRL 规则进行了研究。实验证明，使用本文提出的方法能在解决故障树重复构建问题的同时，不对系统故障的快速定位产生影响。下一步的工作是研究探讨如何利用故障树领域本体和 SWRL 规则对故障树进行定量分析。

参 考 文 献

[1] Arnold F, Belinfante A, Van der Berg F, et al. DFTC$_{ALC}$: A Tool for Efficient Fault Tree Analysis[M]// Computer Safety, Reliability, and Security. Heidelberg: Springer Berlin Heidelberg, 2013: 293-301

[2] Weikum G, Theobald M. From information to knowledge: harvesting entities and relationships form web sources[C]// Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on principles of database system. Indiana: ACM Press, 2010: 65-76

[3] Navigli R, Ponzetto S P. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network[J]. Artificial Intelligence, 2012, 193: 217-250

[4] Cimino J J. High-quality, standard, controlled healthcare terminologies come of age[J]. Methods of information in medicine, 2011, 50(2): 101-104

[5] 葛强，沈国华，黄志球，等. Web 服务中支持本体推理的隐私保护研究[J]. 计算机科学与探索，2013，7(6)：536-544
Ge Qiang, Shen Guo-hua, Huang zhi-qiu, et al. Web Research on Privacy Protection Based on Ontology in Web Service [J]. Journal of Frontiers of Computer Science and Technology, 2013, 7(6): 536-544

[6] 黄凤. 基于描述逻辑的访问控制策略冲突检测方法研究[D]. 南京：南京航空航天大学，2010
Huang Feng. A description logic-based approach for access control policy conflict detection [D]. Nanjing: Nanjing University of Aeronautics and Astronautics, 2010

[7] 钟秀琴，符红光，余莉，等. 基于本体的几何学知识获取及知识表示[J]. 计算机学报，2010，33(1)：167-174
Zhong Xiu-qin, Fu Hong-guang, She Li, et al. Geometry knowledge Acquisition and Representation on Ontology[J]. Chinese Journal of Computers, 2010, 33(1): 167-174

[8] 郑晓洁，张琳. 本体映射中相似度计算的改进[J]. 计算机科学，2013，40(12)：108-112
Zheng Xiao-jie, Zhang Lin. Modification of Similarity Computation in Ontology Mapping [J]. Computer Science, 2013, 40(12): 108-112

[9] 杜小勇，李曼，王珊. 本体学习研究综述[J]. 软件学报，2006，17(9)：1837-1847
Du Xiao-yong, Li Man, Wang Shan. A survey on ontology learning research[J]. Journal of Software, 2006, 17(9): 1837-1847

[10] Noy N F, McGuinness D L. Ontology Development 101: A Guide to Creating Your First Ontology[DB/OL]. (2001-08) [2014-08]. http://protege.stanford.edu/publications/ontology_development/ontology101.pdf

[11] Horrocks I, Patel-Schneider P F, Boley H, et al. SWRL: A semantic web rule language combining OWL and RuleML[OL]. http://xml.coverpages.org/ni2004-05-21-a.html

[12] Boley H B G, Tabet S. Rule Markup Tutorial[DB/OL]. (2005-05) [2014-08]. http://www.ruleml.org/papers/tutorialruleml-20050513.html

[13] W3C. OWL Web Ontology Language Semantics and Abstract Syntax[DB/OL]. (2004-02)[2014-08]. http://www.w3.org/TR/2004/REG-owl-semantics-20040210.html

(上接第 165 页)

[16] 于会，刘尊，李勇军. 基于多属性决策的复杂网络节点重要性综合评价方法[J]. 物理学报，2013，62(2)：20204-9
Yu Hui, Liu Zun, Li Yong-Jun, et al. Key nodes in complex networks identified by multi-attribute decision-making method[J]. Acta Physica Sinica, 2013, 62(2): 20204-9

[17] Fruchterman T M J, Reingold E M. Graph Drawing by Force-directed Placement[J]. Software-practice and Experience, 1991, 21(11): 1129-1164

[18] Holme P, Kim B J, Yoon C N, et al. Attack vulnerability of complex networks [J]. Physical Review E, 2002, 65(5): 56-109

[19] Jeong H, Mason S, Barabási A l, et al. Lethality and centrality in protein networks[J]. Nature, 2001(411): 41-42

[20] Qi Xing-qin, Duval R D, Christensen K, et al. Terrorist Networks, Network Energy and Node Removal A New Measure of Centrality Based on Laplacian Energy[J]. Scientific Research, 2013, 10(4236): 19-31