临床实践指南的自动执行:一种 ASP 方法

黄　皎,张志政

(东南大学计算机科学与工程学院,江苏 南京 211189)

[**摘要**]　临床实践指南是系统化制定的临床指导意见,能够帮助医生和患者在特定的临床场景下选择合适的医疗保健服务. 然而基于文本的指南不便于推广和应用,如何实现指南的自动执行以辅助医生和患者提高医疗质量,已经成为近年来临床决策支持研究的重要内容. 计算机可理解的指南表示语言是此类研究的关键,针对目前对高容变、语义直观的指南表示语言的需求,本文的主要工作是在严格定义了基于指南模型 SDA* 的治疗方案生成过程的基础上,将基于 SDA* 的治疗方案生成直观地转化为求解逻辑程序的回答集.

[**关键词**]　临床实践指南,临床决策支持系统,逻辑程序,知识表示

[**中图分类号**]TP311　[**文献标志码**]A　[**文章编号**]1672-1292(2015)02-0046-09

Automatical Execution of Clinical Practice Guidelines: An ASP Approach

Huang Jiao, Zhang Zhizheng

(School of Computer Science and Engineering, Southeast University, Nanjing 211189, China)

Abstract: Clinical practice guidelines (CPGs), are defined as systematically developed statesments to assist physician and patient in making decisions about appropriate health care for specific circumstances. However, these paper-based guidelines are particularly difficult to be applied into real environment. How to realize the automatical execution of guidelines has become a research focus of clinical decision support. Computer-interpretable guideline modeling language is the key point in these researches. Intuitive semantics will facilitate the users to model CPGs in the language, and elaboration tolerance is required to faciliate the update of the models. In this paper, we present a precise definition of the clinical treatment based on a guideline model SDA* and transform it into a logic programming intuitively.

Key words: clinical practice guidelines, decision-support systems, logic program, knowledge representation

临床实践指南(Clinical Practice Guidelines,CPGs)开发方法科学严谨,临床内容准确可靠,已经成为指导医学研究和临床实践的重要手段. 临床实践指南开发的目的在于规范医疗过程,提高医疗质量,降低医疗成本,减少医疗差错. 然而临床实践指南在临床实践中未能发挥其应有的功效,其推广和应用仍然面临着很多问题:首先,指南一般以文本形式发布,这种基于文本的指南不利于医疗工作者在实践中查阅;其次,现有医学知识海量、且频繁更新,医疗过程复杂,无疑增加医疗工作者的记忆负担;最后,这种基于文本的指南并不能真正实现规范化的临床医疗,缺乏一套相应的规范化机制来促使医疗工作者尽量遵守医疗规范从而减少医疗差错.

随着医院信息化程度的不断提高,为了最大化地发挥临床实践指南的效能,需要将其形式化为计算机可解释的指南(Computer-Interpretable Guidelines, CIGs)[2],开发临床决策支持系统(Clinical Decision Support System,CDSS)[3]以支持指南的自动执行. CIGs 开发的生命周期一般有如下步骤[4]:(1)建立一种 CIG 模型语言;(2)基于模型语言进行知识获取和表示;(3)和电子医疗记录(Electronic Health Records, EHRs)以及工作流进行整合;(4)对 CIGs 进行验证以保证其可靠性;(5)部署 CIGs 到执行引擎并构建基于 CIGs 的应用;(6)CIGs 的维护和更新;(7)CIGs 的共享. 从上述 CIGs 的开发周期可以看出,要实现 CPGs 到 CIGs 的转换,首先需要建立一个指南模型语言,这种语言一方面需要具备高容变性,能表示结构

收稿日期:2014-08-16.
基金项目:国家自然科学基金(60803061)、江苏省自然科学基金(BK2008293)、东南大学科技基金(XJ2008315).
通讯联系人:黄皎,硕士,研究方向:知识表示与推理. E-mail:hjeve1990@ gmail.com

复杂,内容丰富的医学知识,构建可修改、可替换、可扩展的医学知识库,为系统应用层提供数据支持;另一方面要具备高效的推理引擎,作为实现医疗决策功能的核心,有效地实现医学规划、医学诊断、医学问答等功能. 此外,能够接近自然语言,具有直观语义的语言将大大方便 CPGs 到 CIGs.

在国际上公开发布的指南模型有近 20 项[5],比如 Asbru[6]、GLIF3[7,10]、PROforma[8]、EON[9]、SDA*[11,12]等等. 大多数模型都是基于任务网络模型(Task-Network Models,TNMs),可用来表示复杂的、多步骤的临床指南知识,且支持以可视化的方式对指南中的任务序列进行建模. 其中 SDA* 是一种新型的指南模型,它将文本的 CPGs 转换为流程图模型. 针对目前对指南表示语言的需求,本文采用具有高容变[14]、语义自然和高效推理机[15,16]的回答集逻辑程序(Answer Set Prolog,ASP)[13]作为指南表示语言,以 SDA* 作为指南建模工具,提出一种基于 ASP 的声明式的指南表示和治疗方案生成方法.

1　背景知识

1.1　SDA*

SDA* 全称是 State-Decision-Action,表示图中的 3 种类型的节点,紧跟的 * 号表示这些节点可以重复发生多次,它主要用来描述医疗护理中的治疗过程. SDA* 模型的语法和语义主要基于集合论,具体的定义可参考文献[12].

1.1.1　术语集

给定疾病 M,有限集合 T 是 M 在医学领域中的相关术语组成的集合,T 中的术语可进一步分为 3 类:状态术语(State Terms),描述流程中的病人状态;决策术语(Decision Terms),描述流程中的选择性知识;动作术语(Action Terms),描述流程中的医学动作. 这 3 类术语的集合可以有交集,它们的并集即为针对疾病 M 的术语集 T.

带有时序约束的术语又称为时序术语(Temporal Terms),时序约束包括开始时间、结束时间和执行频率. 形式化地,一个术语是一个四元组<*termname*,*start*,*end*,*frequency*>,其中每一元分别表示术语名称,开始时间、结束时间和执行频率. 特别地,没有时序约束的术语,也表示为上述四元组,但是后面 3 个时间约束都为空,也通常简写为术语名称.

1.1.2　元素

SDA* 模型中有 3 类基本的元素,分别为状态(State)节点、决策(Decision)节点、动作(Action)节点. 每个节点都是一个术语子集,状态节点中的术语集是状态术语集的子集,而决策节点有若干分支(Branch),每个分支中的术语集是决策术语集的子集,动作节点的术语集则是动作术语集的子集,表示几个皆需执行的医学动作. 对于某一决策节点 $d=\{D:D_1,D_2,\cdots,D_k,\text{otherwise}\}$,其中 D 是决策节点的术语集,D_i 是决策节点 d 的某个分支术语集($D=D_1\cup D_2\cup\cdots\cup D_k\cup\{\text{otherwise}\}$),分支术语集为{otherwise}的分支称之为决策节点的 otherwise 分支.

1.1.3　连接边

SDA* 模型中的连接边是有向边,连接了上述 3 种类型的元素,连接边相连的 2 个元素分别称为这条边的入元素(in-element)和出元素(out-element). 入元素为动作节点的连接边上也可以带有时序约束,连接边的时序约束是一个二元组<*min*,*max*>,表示动作节点中动作术语所代表的医学动作的延迟或是持续时间.

图 1 是针对具有行走能力的中风病人给出的 SDA* 模型,每个节点内和边上的字符串就是此模型中的术语,圆形节点为状态节点,矩形节点为动作节点,菱形节点为决策节点. 此中风病人模型 M 的状态术语集 $S_M=\{\text{walking\_patient},\text{dispatch}\}$,决策术语集 $D_M=\{\text{walk\_without\_device},$

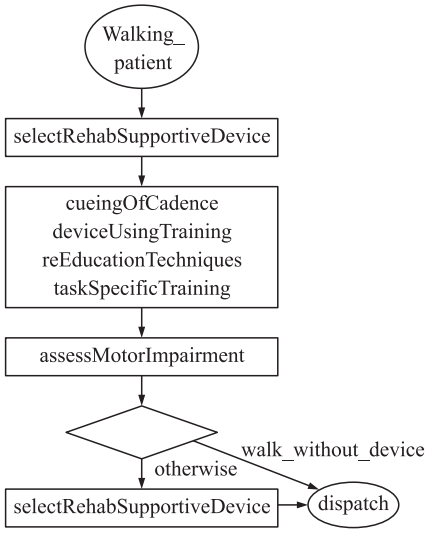

图 1　中风病人的 SDA* 模型

Fig. 1　SDA* model of the Post-Stroke patient

otherwise}, 动作术语集 A_M = {selectRehabSupportiveDevice, cueingOfCadence, deviceUsingTraining, reEducation-Techniques, taskSpecific Training, assessMotorImpair-ment}, 整个模型的术语集 $T_M = S_M \cup D_M \cup A_M$.

1.2 回答集程序

ASP 是基于回答集语义[17]的逻辑编程范式,其强大的非单调推理能力使得其非常适合知识表示与推理. 具体的 ASP 语法和语义可参考文献[18],一个 ASP 程序由以下形式的规则组成:

$$l_0 : -l_1, \cdots, l_m, not\ l_{m+1}, \cdots, not\ l_n. \tag{1}$$

式中:每一个 $l_i, i \in [0 \cdots n]$,是一个文字,具有形式 $p(t)$ 或者 $-p(t)$,p 是谓词(predicate),t 是项(term),not 是缺省否定. 一个没有体部的规则称之为事实,而没有头部的规则称之为约束. 规则(1)读为:如果我们相信 $l_1, \cdots, l_m$ 为真,且没有理由相信 $l_{m+1}, \cdots, l_n$ 为真,则相信 l_0 为真. 而 ASP 程序的回答集就是那些我们相信为真的文字的集合,这个集合可以满足程序中每一条形如(1)所示的规则.

ASP 的应用范围很广,在规划、诊断、知识问答等领域都有应用. 其中,基于 ASP 解决规划问题在文献[18]中作了详细的阐述,经典的规划问题定义如下:

目标(goal):智能 Agent 想要使系统达到某个状态,这个状态中的期望的文字集合均为真.

规划(plan):智能 Agent 为达到目标所做的一系列动作,使得系统从某一个状态迁移到另一个状态,最终到达满足目标的状态.

问题(problem):给定一个确定性动态系统的描述,其初始状态以及目标,需要找到一个规划使系统最终满足目标.

视野(horizon):规划的长度限制. 给定一个规划问题 P,构造一个视野为 n 的 ASP 规划程序 plan(P, n),程序 plan(P,n)的回答集与问题 P 的解决方案之间的关系参考文献[18].

2 基于 SDA* 的医疗方案生成

给定一个 SDA* 模型 M 和病人状态集 C,C 是一些描述病人状态的术语集合,首先给出如下定义:

定义 1(可进入点)　一个状态节点 s 是可进入点,也称之为 C 满足状态节点 s,如果 $s \subseteq C$,记做 $C \vdash s$.

C 满足一个决策节点 $d = \{D : D_1, D_2, \cdots, D_k, \text{otherwise}\}$ 的分支 D_i,如果 $D_i \subseteq C$,记做 $C \vdash d_i$;C 满足决策节点 d 的 otherwise 分支当且仅当对任意 i,$C \vdash D_i$.

定义 2(可选分支)　一个决策节点 $d = \{D : D_1, D_2, \cdots, D_k, \text{otherwise}\}$ 的分支 D_i 是可选分支,即 C 满足分支 D_i,如果 $D_i \subseteq C$,记做 $C \vdash s$. C 满足决策节点 d 的 otherwise 分支当且仅当对任意 i,$C \vdash D_i$.

给定一个 SDA* 模型 M 和病人状态 C,其一个候选治疗方案(candidate treatment)是模型 M 中能够为 C 满足的一条路径(path)上的动作节点的有序集合,亦即一个动作集合的序列. 严格地,我们定义治疗方案的生成如下.

定义 3(候选治疗方案)　给定一个 SDA* 模型 M 和病人状态 C,动作集合序列 $A_1 \cdots A_n$ 是一个候选治疗方案,当且仅当存在一条 M 上的路径 $v_1 \cdots v_m$ 满足:

(1)v_1 是一个可进入点;

(2)如果 v_i 是一个状态节点,且 $1<i<m$,则 $C \vdash v_i$;

(3)如果 v_i 是一个决策节点,且 $1<i<m$,则存在 j 使得 $C \vdash (v_i)_j$;

(4)v_m 是一个不可进入点,或者 v_m 与后继节点之间的边带有时序约束;

对图 1 所示 SDA* 模型,假设病人状态集为{walking_patient},则该模型的路径 P_1 为:

state1{walking_patient}→
action1{selectRehabSupportiveDevice}→
action2{cueingOfCadence, deviceUsingTraining, reEducationTechniques, taskSpecificTraining}→
action3{assessMotorImpairment}→
decision1→
action4{selectRehabSupportiveDevice}→
state2{dispatch}.

由于决策节点 decision1 没有可选分支,所以默认选择 otherwise 分支. 考虑另一病人状态集为{walking_patient,walk_without_device},则该模型的路径 P_2 为:

state1{walking_patient}→
action1{selectRehabSupportiveDevice}→
action2{cueingOfCadence,deviceUsingTraining,reEducationTechniques,taskSpecificTraining}→
action3{assessMotorImpairment}→
decision1→
state2{dispatch}.

注意到上述两种情形中有一个隐含假设:决策节点的决策信息已知. 而对于决策节点的决策信息未知的情况下,其中一种解决方式是输出其所有可能路径.

3　基于 ASP 的 SDA* 执行方法

假设给定一个 SDA* 模型 M,以及病人状态集 C,原始问题是求出其候选治疗方案,也就是求出模型 M 的所有符合条件的路径. 我们将求解候选治疗方案的问题,归结为规划问题,使得规划问题的解即为候选治疗方案. 首先从规划的角度重新定义该问题如下:

目标:到达模型 M 的一个不可进入点或是带有时序约束的连接边.

规划:引入动作 append,将某一个节点追加到路径之中,直到满足上述目标.

问题:将模型 M 看作一个确定的动态系统,每一个节点都有其自己的状态(在路径之中或不在路径之中),找到一个规划使系统达到目标状态.

视野:模型 M 路径长度的上限值.

对于 SDA* 模型 M,视野为 N,以及病人状态集 C,则基于 ASP 的求解程序 $P(M,N,C)$ 由 3 部分组成:模型知识 $P_f(M)$,规划算法 $P_a(N)$,病人状态 P_c. 需要注意的是 $P_a(N)$ 算法独立于特定模型而存在,也就是说对于任意疾病的 SDA* 模型,有相同的求解治疗方案的算法.

3.1　病人状态 P_c

病人状态是一个术语集,可以直接用如下的 ASP 事实来表示:

patient_condition(N,S,E,F).

式中:N 是术语名,(S,E,F)是术语的时序约束,三者值均为 null 时表示该术语是普通术语. 比如病人状态术语集为{walking_patient},则其对应的 ASP 规则为 patient_condition(walking_patient,null,null,null).

考虑到决策节点的决策信息有可能未知,故需要额外的知识表示模型中的某一个决策节点是否未知,用如下形式的 ASP 事实表示某一个决策节点的决策信息处于未知状态:

patient_condition(DId,unknown).

式中:DId 标识某一决策节点,unknown 则表示该决策节点的决策信息是未知的.

3.2　模型知识 $P_f(M)$

模型知识包含了 SDA* 模型图的所有信息,包括 3 种类型的节点、连接边,以及节点内和边上的术语信息.

sda_id(MId).

sda_term(MId,TId,N,S,E,F).

上面两条规则中第一条表示了模型是针对哪一类病症. 而第二条表示模型所包含的术语,MId 表示模型的 id,TId 表示该术语的 id,同样,N 是术语名,(S,E,F)是时序约束,之后不再赘述.

sda_state(MId,SId).

sda_decision(MId,DId).

sda_action(MId,AId).

上面 3 条规则分别标识了模型中的各个节点. 而对于节点所包含的的术语则用谓词 contain 表示,如下面两条规则表示:

contain(MId,EId,TId).

contain(MId,DId,EId,TId).

第一条用于状态节点或是动作节点,第二条则表示决策节点的分支所包含的术语. 最后,对于连接边则用如下的规则表示:

sda_connector(MId,In,Out,Min,Max).

In 为连接边的入元素,Out 为连接边的出元素,(Min,Max)为时序约束,均为 null 表示为普通边.

通过以上介绍的事实规则,就可以将 SDA* 模型所表达的流程图知识描述出来.

由定义 3 可知,SDA* 模型中的候选治疗方案需要用到相关概念,包括可进入点、可选分支,以及结束点(不可进入点,或者与后继节点之间的边带有时序约束的节点),这一部分仍然属于模型相关知识,下面一一给出这些概念的 ASP 表示.

首先是定义 SDA* 模型的元素(状态节点、决策节点和动作节点),如下 3 条规则:

sda_element(MId,SId):-sda_state(MId,SId).

sda_element(MId,DId):-sda_decision(MId,DId).

sda_element(MId,AId):-sda_action(MId,AId).

其次,定义空的状态节点(没有包含任何术语的状态节点),如下 2 条规则:

not_empty_state(MId,SId):-contain(MId,SId,TId).

empty_state(MId,SId):-not not_empty_state(MId,SId).

然后,定义某个状态节点是否是可进入点以及某条决策分支是否是可选分支. 在此之前,定义谓词 allInPC 表示某个状态节点包含的术语集以及决策节点的某个分支所包含的的术语集包含于病人状态术语集.

```
patient_term(N):-patient_condition(N,S,E,F).
exitNotInPC(MId,SId):-sda_state(MId,SId),contain(MId,SId,TId),
        sda_term(MId,TId,N,S,E,F),not patient_term(N).
allInPC(MId,SId):-sda_state(MId,SId),not exitNotInPC(MId,SId).
exitNotInPC(MId,Did,EId):-contain(MId,DId,EId,TId),sda_term(MId,TId,N,S,E,F),
        not patient_term(N).
allInPC(MId,DId,EId):-contain(MId,DId,EId,TId),sda_term(MId,TId,N,S,E,F),
        not exitNotInPC(MId,Did,EId).
```

而下面的规则是真正定义一个状态节点是否是一个可进入点. 与 SDA* 模型中的定义稍有不同是这里假设可进入点不是任何连接边的出元素,这样保证了路径的开始点是没有入边的节点.

```
feasible_state(MId,SId):-empty_state(MId,SId).
feasible_state(MId,SId):-allInPC(MId,SId).
not_start_state(MId,SId):-sda_state(MId,SId),sda_connector(MId,EId,SId,Min,Max).
start_state(MId,SId):-sda_state(MId,SId),not_start_state(MId,SId).
feasible_entry_point(MId,SId):-feasible_state(MId,SId),start_state(MId,SId).
```

接着是可选分支的定义,并单独定义了 otherwise 分支,这里把 otherwise 视为一个术语.

```
feasible_branch(MId,DId,EId):-allInPC(MId,DId,EId).
otherwise_branch(MId,DId,EId):-contain(MId,DId,EId,TId),
        sda_term(MId,TId,otherwise,null,null,null).
```

由于需要判断某个决策节点是否存在可选分支,在没有可选分支的情况默认选择 otherwise 分支,故定义了 exit_feasible_branch 和 no_feasible_branch 这 2 个谓词分别表示存在可选分支和不存在可选分支:

```
exit_feasible_branch(MId,DId):-contain(MId,DId,EId,TId),feasible_branch(MId,DId,EId).
no_feasible_branch(MId,DId):-contain(MId,DId,EId,TId),not exit_feasible_branch(MId,DId).
```

最后,则需要定义模型的结束点,即到达非可进入点或是达到时序边时结束,如下两条规则所示:

finish_point(MId,SId):-sda_state(MId,SId),not feasible_state(MId,SId).

finish_point(MId,InId):-sda_element(MId,InId),sda_connector(MId,In,Out,Min,Max),
Min ! =null,Min>0.

至此,SDA* 模型中所需要的概念都以声明性的方式定义完毕,可以看出此方式清晰简单,下面给出其规划算法.

3.3 规划算法 $P_a(N)$

把一个 SDA* 模型看成一个动态系统,首先给出其视野以限制规划的最大长度,如下面 3 条规则所示:

```
#const n=100.
step(0…n).
#domain step(I).
```

#const 和#domain 是由推理机 clingo[19] 提供的功能,前者可以允许以不同的视野来运行程序,后者避免了在规则中重复声明变量. 这里默认视野长度为 100.

定义系统中节点的流属性,

```
fluent(inertial,in_trace(MId,EId)):-sda_element(MId,EId).
fluent(defined,possible_in_trace(MId,EId)):-sda_element(MId,EId).
```

如上述两条规则所示,给每个元素节点定义了两种类型的流,一种是惯性流 in_trace,表示在步骤 I 时,某个节点是否已在路径之中,"惯性流"的含义是在没有受到任何影响的情况下,对象在步骤 I 的状态将会保持到步骤 I+1,规则(2)和(3)描述了这种现象,称之为惯性定律(intertia axiom)[18].

```
holds(F,I+1):-fluent(inertial,F),holds(F,I),not-holds(F,I+1).        (2)
-holds(F,I+1):-fluent(inertial,F),-holds(F,I),not holds(F,I+1).       (3)
```

另一种是定义流 possible_in_trace,表示在步骤 I 时,某个节点是否有可能在路径之中,定义流满足封闭世界假设(closed world assumption)[18],规则(4)描述了此假设.

```
-holds(F,I):-fluent(defined,F),not holds(F,I).                        (4)
```

在描述完节点的流属性之后,需要定义可以改变动态系统状态的动作,这里只引入一个动作 append,用于追加节点到路径之中:

```
action(append(MId,EId)):-sda_element(MId,EId).
```

在定义了流和动作之后,可以给出系统的域描述,域描述共由 8 条规则组成,其中(5)~(10)是因果规则(causal laws),(11)是一条约束规则(impossible rules).

```
holds(in_trace(MId,EId),I+1):-occurs(append(MId,EId),I).              (5)
holds(possible_in_trace(MId,Out),I+1):-occurs(append(MId,In),I),       (6)
    sda_connector(MId,In,Out,Min,Max),sad_action(MId,In).
holds(possible_in_trace(MId,Out),I+1):-occurs(append(MId,In),I),       (7)
    sda_connector(MId,In,Out,Min,Max),feasible_state(MId,In).
0{holds(possible_in_trace(MId,Out),I+1):sda_connector(MId,In,Out,Min,Max)}1:-
    sda_decision(MId,In),occurs(append(MId,In),I),patient_condition(In,unknown).   (8)
0{holds(possible_in_trace(MId,Out),I+1):feasible_branch(MId,In,Out)}1:-sda_decision(MId,In),
    occurs(append(MId,In),I),not patient_condition(In,unknown).        (9)
holds(possible_in_trace(MId,Out),I+1):-no_feasible_branch(MId,In),otherwise_branch(MId,In,Out),
    occurs(append(MId,In),I),not patient_condition(In,unknown).        (10)
-occurs(append(MId,EId),I):--holds(possible_in_trace(MId,EId),I).      (11)
```

规则(5)表示在步骤 I 对任一节点发生 append 动作时,则此节点在步骤 I+1 会变成 in_trace 状态;规则(6)和(7)表示在步骤 I 对某一动作节点或是可达状态节点发生 append 动作,则其邻接节点在步骤 I+1 会变成 possible_in_trace 状态;规则(8)表示在决策节点的决策信息未知的情况下,从所有分支中选择一条分支;规则(9)表示在步骤 I 对某一决策节点发生 append 动作时,且在决策节点的决策信息已知的情况下,

其可选分支的出元素会变成 possible_in_trace 状态(这里假设决策节点是 XOR 节点,但多个回答集表示有多条符合条件的路径可供选择);而规则(10)则表示当某一决策节点没有可选分支时,且在决策节点的决策信息未知的情况下,会默认选择 otherwise 分支,其 otherwise 分支的出元素将处于 possible_in_trace 状态;最后,约束规则(11)表示若某个节点在步骤 I 不是 possible_in_trace,则不能对此节点发生 append 动作.

基于上述系统域描述,用如下规则指定规划方式:

```
success:-goal(I),I<=n.
:-not success.
1{occurs(Action,I):action(Action)}1:-not goal(I),I<n.
```

上面 3 条规则表示在没有达到目标之前且未超越视野 n 时,可以发生一个动作,且一旦达到目标就停止规划.

最后,需要定义规划的目标以及系统的初始状态,下面这条规则表示当结束点在路径之中时就达到目标状态.

```
goal(I):-finish_point(MId,EId),holds(in_trace(MId,EId),I).
```

还需要设定系统的初始状态,只有可进入点处于 possible_in_trace 状态,所有节点都不处于 in_trace 状态:

```
holds(possible_in_trace(MId,EId),0):-feasible_entry_point(MId,EId).
-holds(in_trace(MId,EId),0):-sda_element(MId,EId).
```

这样,结合病人状态 P_c,模型知识 $P_f(M)$,独立的规划算法 $P_a(N)$ 组成的一个 ASP 程序就可以求解出符合条件的路径.

4　系统设计

基于 CPGs 的临床决策支持系统架构如图 2 所示.

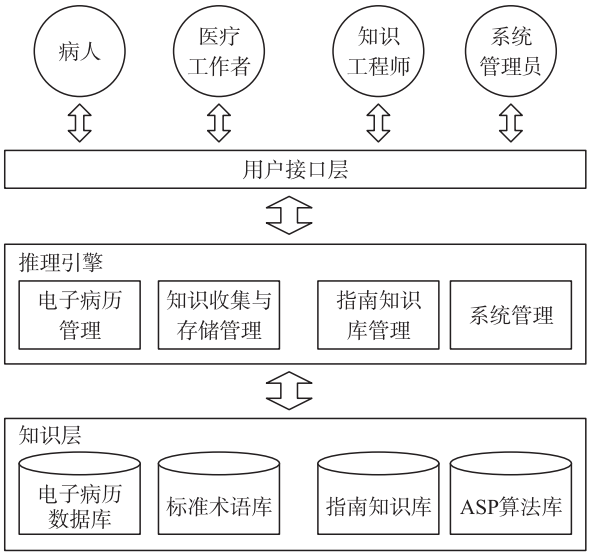

图 2　临床决策支持系统架构图

Fig. 2　The architecture of CDSS

系统面向的用户分为 4 种,病人和医疗工作者可以获得系统提供的基于 CPGs 的各种服务,知识工程师利用系统工具构建指南知识库,系统管理员则进行一些系统支持. 推理引擎则是系统核心模块,主要包含 4 个部分:(1)电子病历管理,获取和管理病人信息;(2)知识收集与存储管理,基于术语库进行知识映射从而统一知识形式;(3)指南知识库管理,构建基于 CPGs 的知识库;(4)系统管理,系统用户及权限管理等. 系统底层知识库主要包括 4 种:(1)电子病历数据库,存储了病人信息;(2)标准术语库,用于统一知识形式;(3)指南知识库,存储指南知识;(4)ASP 算法库,针对不同应用场景的 ASP 推理算法.

为了进行实验分析,将第二部分给出的算法应用到图 1 中针对中风病人的 SDA* 模型,假定病人状态为如下 ASP 事实:

```
patient_condition(walking_patient,null,null,null).
```

应用现有的高效开源的 clingo[19] 推理引擎,可得到如图 3 所示的回答集(这里略去了基本事实,只显示谓词是 occurs 的文字以分析其规划路径).

回答集显示了整个规划路径,根据病人状态和状态节点中的术语可知,state_id1 是该模型的可进入点,首先被追加到路径之中,紧接着是 3 个动作节点 action_id1、action_id2、action_id3,而到达决策节点 decision_1 时,需要根据病人状态判断选择哪一个分支,由于没有任何可选分支,所以默认选择 otherwise 分支,因此 action_id4 被追加到路径之中,最后到达终止节点 state_id2 完成此次规划,可以看出规划路径和

SDA* 执行语义一致(符合第一部分给出的路径 P_1). 考虑另一种不同的病人状态,表述为如下 ASP 事实:

patient_condition(walking_patient,null,null,null).

patient_condition(walk_without_device,null,null,null).

则对应的回答集如图 4 所示.

```
Answer : 1
occurs (append (sda_id1, state_id2), 6)
occurs (append (sda_id1, action_id5), 5)
occurs (append (sda_id1, decision_id1), 4)
occurs (append (sda_id1, action_id3), 3)
occurs (append (sda_id1, action_id2), 2)
occurs (append (sda_id1, action_id1), 1)
occurs (append (sda_id1, state_id1), 0)

Models  : 1
Time    : 0.000
Prepare : 0.000
Prepro. : 0.000
Solving : 0.000
```

图 3　中风病人模型的回答集 1

Fig. 3　The answer sets of Post-Stroke Model-1

```
Answer : 2
occurs (append (sda_id1, state_id2), 5)
occurs (append (sda_id1, decision_id1), 4)
occurs (append (sda_id1, action_id3), 3)
occurs (append (sda_id1, action_id2), 2)
occurs (append (sda_id1, action_id1), 1)
occurs (append (sda_id1, state_id1), 0)

Models  : 2
Time    : 0.000
Prepare : 0.000
Prepro. : 0.000
Solving : 0.000
```

图 4　中风病人模型的回答集 2

Fig. 4　The answer sets of Post-Stroke Model-2

同样,在到达决策节点之前的规划动作与前一实验一致. 这里由于病人状态的不同使得决策节点有可选分支,故 otherwise 分支被废止,转而选择可选分支,直接到达终止节点 state_id2,回答集给出的规划路径同样符合第一部分给出的路径 P_2. 而从这些规划路径中可以获取其中的动作节点以及节点所包含的术语,而这些动作便构成了针对具体病症的一个治疗过程.

最后,考虑决策节点的决策信息未知的情况,即病人状态信息如下:

patient_condition(walking_patient,null,null,null).

patient_condition(decision1,unknown).

则得出回答集如图 5 所示,可以看到在决策节点未知的情况下,输出了所有的路径,而每一条路径对应一个回答集.

由于篇幅有限,上述算法和实验未考虑时序术语.

```
Answer : 1
occurs (append (sda_id1, state_id2), 6)
occurs (append (sda_id1, action_id5), 5)
occurs (append (sda_id1, decision_id1), 4)
occurs (append (sda_id1, action_id3), 3)
occurs (append (sda_id1, action_id2), 2)
occurs (append (sda_id1, action_id1), 1)
occurs (append (sda_id1, state_id1), 0)

Answer : 2
occurs (append (sda_id1, state_id2), 5)
occurs (append (sda_id1, decision_id1), 4)
occurs (append (sda_id1, action_id3), 3)
occurs (append (sda_id1, action_id2), 2)
occurs (append (sda_id1, action_id1), 1)
occurs (append (sda_id1, state_id1), 0)

Models  : 1
Time    : 0.000
Prepare : 0.000
Prepro. : 0.000
Solving : 0.000
```

图 5　中风病人模型的回答集 3

Fig. 5　The answer sets of Post-Stroke Model-3

5　结论

本文提出了一种基于 ASP 的方法来实现临床实践指南到计算机可解释的指南的转化以支持自动执行和推理. 提出用 ASP 事实表示 SDA* 模型,用基于规划的 ASP 算法求解治疗方案. 此方法与已有模型和系统相比有如下优点:(1)由于 ASP 本身具有的声明型编程特点,使得整个求解过程简单清晰,只需要将相关问题描述清楚,便可以得出其答案,这样使得 CIGs 的开发更为高效,降低开发和维护成本;(2)ASP 强大的知识表示能力,一方面可以表示不确定性和不完全知识(比如 unknown 信息,otherwise 分支),另一方面又不仅仅局限于目前的 SDA* 模型,还可以表示其他模型,以达到指南的可扩展和可共享;(3)本文的算法得到的结果可以用于其他应用,比如进行并发症(多个疾病模型存在动作冲突)的求解,或是进行结果的一致性验证(检查是否符合特定模型),以及用作医疗教育等等,而这些应用也可以用独立的 ASP 算法求解,还有待进一步研究.

[参考文献](References)

[1] Field, Marilyn J, Kathleen N L, et al. Guidelines for Clinical Practice: from Development to Use [M]. Washington, DC: National Academies Press, 1992.

[2] Peleg M, Tu S, Bury J, et al. Comparing computer-interpretable guideline models: a case-study approach[J]. Journal of the American Medical Informatics Association, 2003, 10(1): 52-68.

[3] Mendonça E A. Clinical decision support systems: perspectives in dentistry[J]. Journal of Dental Education, 2004, 68(6): 589-597.

[4] Peleg M. Computer-interpretable clinical guidelines: A methodological review[J]. Journal of Biomedical Informatics, 2013, 46(4): 744-763.

[5] 吴彬飞. 临床指南知识表达和应用方法研究[D]. 杭州: 浙江大学生物医学工程与仪器科学学院, 2010.
Wu Binfei. Study of knowledge representation and application methods for dinical practice practice guidelines[D]. Hangzhou: Zhejiang College of Biomedical Engineering and Instrument Science of Zhejiang University, 2010.

[6] Shahar Y, Miksch S, Johnson P. The Asgaard project: a task-specific framework for the application and critiquing of time-oriented clinical guidelines[J]. Artificial Intelligence in Medicine, 1998, 14(1): 29-51.

[7] Boxwala A A, Peleg M, Tu S, et al. GLIF3: a representation format for sharable computer-interpretable clinical practice guidelines[J]. Journal of Biomedical Informatics, 2004, 37(3): 147-161.

[8] Sutton D R, Fox J. The syntax and semantics of the Proforma guideline modeling language[J]. Journal of the American Medical Informatics Association, 2003, 10(5): 433-443.

[9] Musen M A, Tu S W, Das A K, et al. EON: a component-based approach to automation of protocol-directed therapy[J]. Journal of the American Medical Informatics Association, 1996, 3(6): 367-388.

[10] Peleg M, Boxwala A A, Bernstam E, et al. Sharable representation of clinical guidelines in GLIF: relationship to the Arden Syntax[J]. Journal of Biomedical Informatics, 2001, 34(3): 170-181.

[11] Campana F, Moreno A, Riaño D, et al. K4care: Knowledge-based Homecare E-services for an Ageing Europe[M]. Basel: Birkhäuser, 2008: 95-115.

[12] Riano D. The SDA model: a set theory approach[C]//Twentieth IEEE International Symposium on Computer-Based Medical Systems. Slovenia: IEEE Computer Society, 2007: 563-568.

[13] Lifschitz V. What is answer set programming? [C]//The 23rd National Conference on Artificial Intelligence. Chicago: AAAI, 2008, 8: 1 594-1 597.

[14] McCarthy J. Elaboration tolerance[C]//Common Sense. London: Citeseer, 1998: 98.

[15] Faber W, Pfeifer G, Leone N, et al. Design and implementation of aggregate functions in the DLV system[J]. TPLP, 2008, 8(5/6): 545-580.

[16] Gebser M, Kaufmann B, Neumann A, et al. Clasp: A Conflict-driven Answer Set Solver[M]. Heidelberg: Springer Berlin, 2007: 260-265.

[17] Gelfond M, Lifschitz V. The stable model semantics for logic programming[C]//ICLP/SLP. Cambridge: MIT Press, 1988, 88: 1 070-1 080.

[18] Gelfond M, Kahl Y. Knowledge Representation, Reasoning, and the Design of Intelligent Agents: The Answer-set Programming Approach[M]. Cambridge: Cambridge University Press, 2014.

[19] Gebser M, Schaub T, Thiele S. Gringo: A New Grounder for Answer Set Programming[M]//Logic Programming and Nonmonotonic Reasoning. Heidelberg: Springer Berlin, 2007: 266-271.

[责任编辑: 黄　敏]