

The SDA Model v1.0: a Set Theory approach

David Riaño

Research Group on Artificial Intelligence

Rovira i Virgili University

28/02/2007

Tarragona, Spain

Table of Contents

1	Introduction	3
2	Antecedents	5
2.1	Asbru	5
2.2	PROforma	6
2.3	EON	7
3	The SDA* Model: Syntax and Semantics	9
3.1	Formal description	10
3.1.1	The Universe of Discourse	11
3.1.2	Elements	11
3.1.3	Connectors	14
3.2	Sequences and cycles	16
3.3	Non-Determinism	18
3.4	Time	19
3.5	Parallelism	20
4	Construction and execution of health procedures with the SDA* Model	22
4.1	Abstract data type SDA* procedure	22
4.2	Textual representation of SDA* procedures	23
4.3	Execution of SDA* procedures	24
4.4	Examples	25
4.4.1	Representing partial knowledge	26
4.4.2	CSI's Hypertension Diagnosis and Treatment	27
4.4.3	Comprehensive Assessment K4CARE Procedure	28
4.4.4	The use of Antidepressant Medication in the Elderly	30
4.4.5	Management of Depression with Cognitive Impairment	31
4.4.6	Management of Depression with Dementia	32
4.4.7	Suicide: Risk of Assessment and Management	33
5	Conclusions and Acknowledgements	34
6	References	35
	APPENDIX: Schema sda.xsd	36

1 Introduction

K4CARE (IST-2004-026968: Knowledge Based Homecare eServices for an Ageing Europe) is a Specific Targeted Research or Innovation Project funded within Sixth Framework Programme of the European Commission (<http://ec.europa.eu/research/fp6/>) that brings together 13 academic and industrial institutions from seven countries for a period of three years starting March 2006.

K4CARE addresses the problem of the care of **chronic disabled patients at home** that, in modern societies, involves life long treatment under continuous expert supervision that saturate European national health services and increase related costs.

The project aims at combining the healthcare and the ICT experiences of several western and eastern EU countries to create, implement, and validate a **knowledge-based healthcare model** for the professional assistance to senior patients at home. This new Healthcare Model for home care will contribute to achieve a European standard supported by the new technologies that improves the efficiency of the care services for all the citizens in the enlarged Europe.

In the K4CARE project, procedures, formal intervention plans and individual intervention plans are the basic structures to represent health care procedural knowledge (or *know how*). In this setting, a **procedure** is described as an implementation of a health care service by means of a combination of actions [2]. For example, the steps that configure a blood analysis or a comprehensive assessment, or the health care activities involved in comprehensive assessment¹. **Formal Intervention Plans (FIPs)** are defined as formal structures representing the healthcare procedures to assist patients suffering form particular ailments or diseases. They contain indications to all the actors involved in the care process (i.e. healthcare professionals, patients and relatives, etc.) in order to provide the best coordinated and effective action plan. A FIP on *hypertension*, for instance, provides the indications of how to act with a hypertensive patient in general (see Figure 4). FIPs are general structures that have to be adapted to the particularities of a patient before it is actionable an applicable to this patient. In the K4CARE project the structure resulting from this adaptation is called **individual intervention plan (IIP)**.

In the K4CARE Project, these three structures are used to:

- represent the professional worldwide existing "know-how" knowledge within the K4CARE platform;
- guide the K4CARE services the system offers to the professional users;
- make explicit the way Home Care (HC) must be provided in a growing ageing EU;
- offer a knowledge representation frame in which the new machine learning techniques developed in the project make explicit the knowledge about HC interventions implicit in

¹ *Comprehensive assessment* of a patient comprises Multi-Dimensional Evaluation plus Clinical Assessment and Physical Examination (integrating the medical side) and Social Needs and Social Network Assessment (integrating the social side). It is fully described in section 4.4.3.

the Electronic Health Care Record (these are learned from the procedures regarding past patients stored in the system);

- offer a representation frame in which procedural knowledge about “pure” pathologies can be integrated in complex or co-morbid pathologies;
- personalize the care to particular patients, having into account their specific characteristics;
- develop a family of FIPs representing procedural knowledge about the treatments of the syndromes targeted in the K4CARE project;
- adapt to a common representation several clinical guidelines already published by international healthcare organizations as the *National Library of Medicine* and the *National Guideline Clearinghouse* in the USA, the *New Zealand Guidelines Group*, the *Scottish SIGN*, etc.;
- use knowledge engineering methods to create new formal representations for conditions and diseases relevant in the project that do not have any trustable treatment published or known. These FIPs will integrate the experiences in the treatment of such cases by all the healthcare partners of the K4CARE consortium, and
- exploit the data in the EHC in order to induce general FIPs from individual patient treatments throughout a long time period.

This paper introduces **the SDA* Model v1.0** as an effective framework to formalize procedures, FIPs, and IIPs in the K4CARE project. First, section 2 will broadly introduce three languages capable of representing FIPs: Asbru, PROforma, and EON; together with some drawbacks that justify the deployment of the new SDA* model. In section 3, the SDA* model is introduced in a formal way. Section 4 contains the proposed interface to construct and execute procedural knowledge under the SDA* model, and also a list of examples. Section 5 contains the conclusions of the work and the acknowledgements. Section 6 displays the list of references.

The document contains an Appendix describing the XML Schema of the SDA* model.

2 Antecedents

This section introduces three successful approaches to represent Clinical Practice Guidelines. Most of the content of this section is taken literally from [1] and from the websites of the three languages discussed. The languages are described in three separate subsections, each one containing a brief history of the language, the recommended papers describing the language, the elements for structural description and time management, and the drawbacks when they are considered to represent FIPs in the K4CARE project.

2.1 Asbru

The ***Asgaard Project*** (www.asgaard.tuwien.ac.at) is a project of the Information Engineering Group, a part of the Information and Software Engineering Group (IFS) of the Institute of Software Technology and Interactive Systems (ISIS) at the Faculty of Informatics of the Vienna University of Technology in collaboration with the Austrian Research Institute for Artificial Intelligence (OFAI), and the Stanford Medical Informatics at Stanford University. The aim of the project is to design a set of tasks that support the design and the execution of skeletal plans by a human executing agent other than the original plan designer. Skeletal plans are a powerful way to reuse existing domain-specific procedural knowledge, but leave room for execution-time flexibility to achieve particular goals. Asbru is the language to represent skeletal plans in Asgaard.

Recommended publications: historical paper [9], formal introduction to the language and time management [7], user's guide [10].

Asbru is the time-oriented, intention-based, skeletal plan-specification representation language in the Asgaard project to represent clinical guidelines and protocols in XML. Asbru can be used to express clinical protocols as skeletal plans that can be instantiated for every patient. It was designed specific for the set of plan-management tasks. Asbru enables the designer to represent both the prescribed actions of a skeletal plan and the knowledge roles required by the various problem-solving methods for performing the intertwined supporting subtasks.

Asbru distinguishes between a declarative data abstraction part and a procedural hierarchy of plans. The data abstraction specification consists of a set of parameter definitions. Each of the plans in the plan hierarchy consists of a name, a set of arguments, a time annotation (representing the temporal scope of the plan), preferences, intentions, conditions, effects, and plan body. Only name and plan body are mandatory.

Arguments are values passed from the invoking or calling plan (called parent) to the invoked or called plan (called child); ***preferences*** describe the costs, resource constraints, and responsible actor; ***intentions*** are high-level goals of the plan represented by temporal patterns of actions and states that should be maintained, achieved or avoided; ***conditions*** mediate the changes between plan states whose normal evolution is (a) wait for the plan to be considered, (b) fulfill the filter precondition, (c) become activated, (d) complete the plan unless the abort or the suspend condition

is fulfilled first, in which case the plan becomes aborted or delayed until a reactivate condition holds, and (d) terminate the plan; **effects** describe the possible consequences of plans as a relationship between plan arguments and measurable parameters by means of mathematical functions or in a qualitative way, and **body** contains set of plans to be executed in a particular way. Asbru distinguishes among four types of plans: in sequence (one after the other), in parallel (simultaneously), in any order (only one at a time), and unordered (anyone at a time).

An important part in specifying the complex temporal aspects of skeletal plans are *time annotations*. As Figure 1 schematizes, **time annotation** specifies four points in time relative to a reference point (which can be a specific or abstract point in time, or a state transition of a plan): the earliest starting shift (ESS), latest starting shift (LSS), earliest finishing shift (EFS) and latest finishing shift (LFS). Two durations can also be defined: The minimum duration (MinDu) and maximum duration (MaxDu). Together, these data specify the temporal constraints within which an action must take place, or a condition must be fulfilled for a condition to trigger.

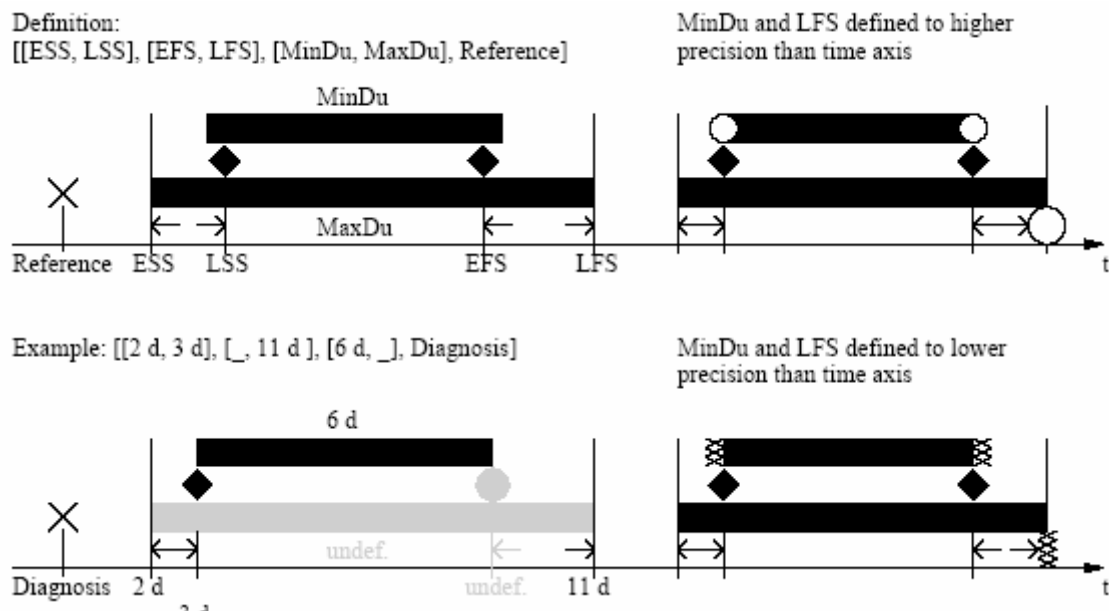


Figure 1. Asbru temporal model.

Asbru is one of the most complete languages to represent procedural knowledge in medicine. However it is not appropriate for the K4CARE project purposes: on the one hand it is defined as a task-based framework and not as a language for representing FIPs, therefore the knowledge it represents is not intuitive to health care professionals. On the other hand, the ability to exploit all the capabilities of Asbru requires a training that actors in the K4CARE model are not expected to fulfill. These are the two main reasons that make Asbru not appropriate in the K4CARE project.

2.2 PROforma

PROforma (www.acl.icnet.uk/lab/proforma.html) was developed by the Advanced Computation Laboratory of the Cancer Research UK. It is a formal knowledge representation language capable of capturing the structure and content of a clinical guideline in a form that can be interpreted by a computer. The language forms the basis of a method and a technology for developing and

publishing executable clinical guidelines. Applications built using PROforma software are designed to support the management of medical procedures and clinical decision making at the point of care.

Recommended publications: historical paper [4], formal introduction to the language [1].

In PROforma, a guideline application is modelled as a set of tasks and data items. The notion of a task is central - the PROforma task model (Figure 2) divides from the keystone (generic task) into four types: plans, decisions, actions and enquiries.

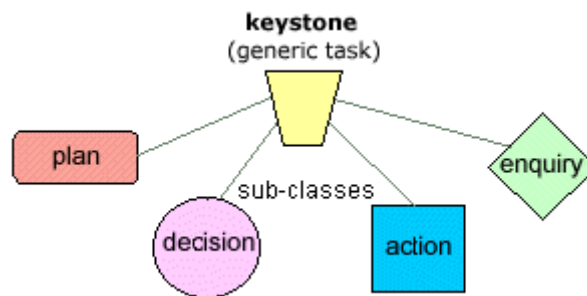


Figure 2. The PROforma task model

Plans are the basic building blocks of a guideline and may contain any number of tasks of any type, including other plans. **Decisions** are taken at points where options are presented, e.g. whether to treat a patient or carry out further investigations. **Actions** are typically clinical procedures (such as the administration of an injection) which need to be carried out. **Enquiries** are typically requests for further information or data, required before the guideline can proceed.

A particularity of the PROforma language is that none temporal model for managing time in medicine is explicitly published.

PROforma is an intuitive language to represent procedural knowledge in medicine. However, it lacks of an explicit representation of temporal constraints which is considered a relevant aspect in the K4CARE project. PROforma is, therefore, not appropriate to the interest of that project.

2.3 EON

The **EON Project** (smi-web.stanford.edu/projects/eon/) at Stanford Medical Informatics was interrupted in 2003 and partially continued with the **SAGE Project** (www.sageproject.net) that integrates experiences from the GLIF3 language. The first project created an architecture made up of a set of software components and a set of interfaces that developers can use to build robust decision-support systems that reason about guideline-directed care. This architecture is based in middleware components (reusable, embeddable software modules) such as a temporal database mediator for handling requests of time-dependent data from a patient database, domain models for multiple clinical specialties, a generic and extensible ontology for modeling clinical guidelines and protocols, an eligibility-determination server, a protocol-based therapy planner; and a mediator for explaining and visualizing the behavior of other EON components.

The **EON guideline modelling and execution system** forms part of the above mentioned EON architecture. It includes an extensible suite of models to represent parts of a clinical practice guideline, domain ontologies, a view of patient data (virtual medical record), and other entities (e.g. those that define roles in an organization). The guideline model (called the Dharma model) defines guideline knowledge structures such as eligibility criteria, abstraction definitions, guideline algorithm, decision models, and recommended actions. The EON guideline execution system obtains patient data through a specified temporal database manager or from user input, and then generates recommendations according to the contents of the specific guideline. The encoding of EON guidelines is done in the **Protégé** knowledge-engineering environment. In the EON guideline model, conditional goals (e.g. if patient is diabetic, the target blood pressures are 135/80) are associated with guidelines and sub-guidelines. The guideline algorithm is represented as a set of scenarios, action steps, decisions, branches, and synchronisation steps connected by a "followed-by" relation. **Scenarios** represent partial characterizations of the state of a patient. **Action steps** describe sets of (instantaneous) action specifications that should be carried out. **Decisions** represent choices from a set of competing alternatives. EON defines two subclasses of decisions: decisions resolved by if-then-else conditions and decisions that require making a heuristic choice from a set of pre-enumerated alternatives. **Branches** represent concurrent action sequences that can converge by means of **synchronization steps**. This EON guideline model appears in Figure 3 together with the EON temporal model that structures the time entities in time points, durations, and time intervals.

A **time point** is an instant time that can be an exact date (definite time point), a vague date (fuzzy time point) or a time with respect to the present time as the concepts of "today", "now", etc. (relative time point). A **duration** is a quantity of time that can be definite (e.g. 5 days) or fuzzy (e.g. 5 days more or less). A **time interval** is the time between two time points.

The guideline model and the temporal model interact through the concept of **criterion**.



Figure 3. Guideline Model and Time model in EON.

EON makes explicit the separation between the representation of FIPs as clinical algorithms in Protégé, from the tools deployed to interpret and use these representations. Unfortunately, the interruption in the development and progressive upgrade of EON causes the product to be in an uncertain dead end situation that affects the confidence of the K4CARE project in this language, in spite of EON's adequacy to the FIP representation requirements in the project.

3 The SDA* Model: Syntax and Semantics

SDA stands for State-Decision-Action, SDA* (SDA star) represents the repetition of states, decisions and actions in order to describe health care procedural knowledge as, for example, K4CARE procedures, FIPs, or IIPs. In the SDA* model, **states** are used to describe patient conditions, situations, or statuses that deserve a particular course of actions which is totally or partially different from the actions to be followed when the patient is in other state. It provides a response to the fact that a disease, ailment, pathology, or syndrome can present alternative degrees of evolution whose treatment must be distinguished. **Decisions** in the SDA* model capture the need of procedural knowledge to represent alternative options whose selection depends on the available information about the patient. In this sense, decisions are able to unify in a single representation of the procedural knowledge alternative courses of actions that have to be applied to patients that meet different conditions. Unlike states, decisions are not intended to make the degree of evolution of a disease explicit, but to orientate a general purpose treatment to the particular characteristics of the patient; for example in the treatment of hypertension, *high-blood-pressure* is a patient condition that may deserve a special treatment and, therefore, it should be represented as a state. On the contrary, in the treatment of cardiac insufficiency, the patient condition *high-blood-pressure* provides information which is relevant to adapt the treatment, but not to decide on the treatment as a whole, which is based on other conditions as *structural-heart-disease* or *prior-heart-problems*. So in cardiac insufficiency, *high-blood-pressure* should be taken as a decision. Finally, **actions** are the proper treatment steps in the SDA* procedural knowledge that are performed according to the preceding decisions.

States, decisions, and actions are combined to form a joined representation of how to deal with a particular health care situation (e.g. a therapy). For example, Figure 4 depicts the FIP that was published by the Institute for Clinical Systems Improvement (www.icsi.org) to diagnose and treat hypertension. It is based on the following indications:

- i. Patients in the FIP can be in four alternative states:
 - a) Screening and identification of elevated BP in patients with diabetes, chronic kidney disease, heart failure, or CAD (FIP element #1).
 - b) Initial assessment completed; i.e. evaluated, accurately staged, and complete risk assessed (FIP element #3).
 - c) Hypertension is suspected to be caused by secondary causes (FIP element #5).
 - d) Hypertension is under control and a continuing care must start (FIP element #12).
- ii. The process is based on three yes-no decisions (one of them appearing twice in the FIP):
 - a) Is a second cause of hypertension suspected (FIP element #4)?
 - b) Is the blood pressure at goal; i.e. within normality limits (FIP elements #7 and #9)?
 - c) Is it a resistant hypertension; i.e. have we fail to achieve a normal BP despite the use of a rational triple-drug regimen in optimal doses (FIP element #10)?
- iii. The actions proposed for the diagnosis and treatment are:
 - a) Confirm hypertension on the initial visit, plus two follow-up visits with at least two BP measures at each visit; following standardized BP measurement techniques, including out of office or home blood pressure measurements (FIP element #2).
 - b) Consider a thiazide-type diuretic as initial therapy in most patients with uncomplicated hypertension (FIP element #6).
 - c) For many patients, two or more drugs in combination may be needed to reach hypertension goals (FIP element #8).
 - d) Refer to hypertension consultation (FIP element #11)

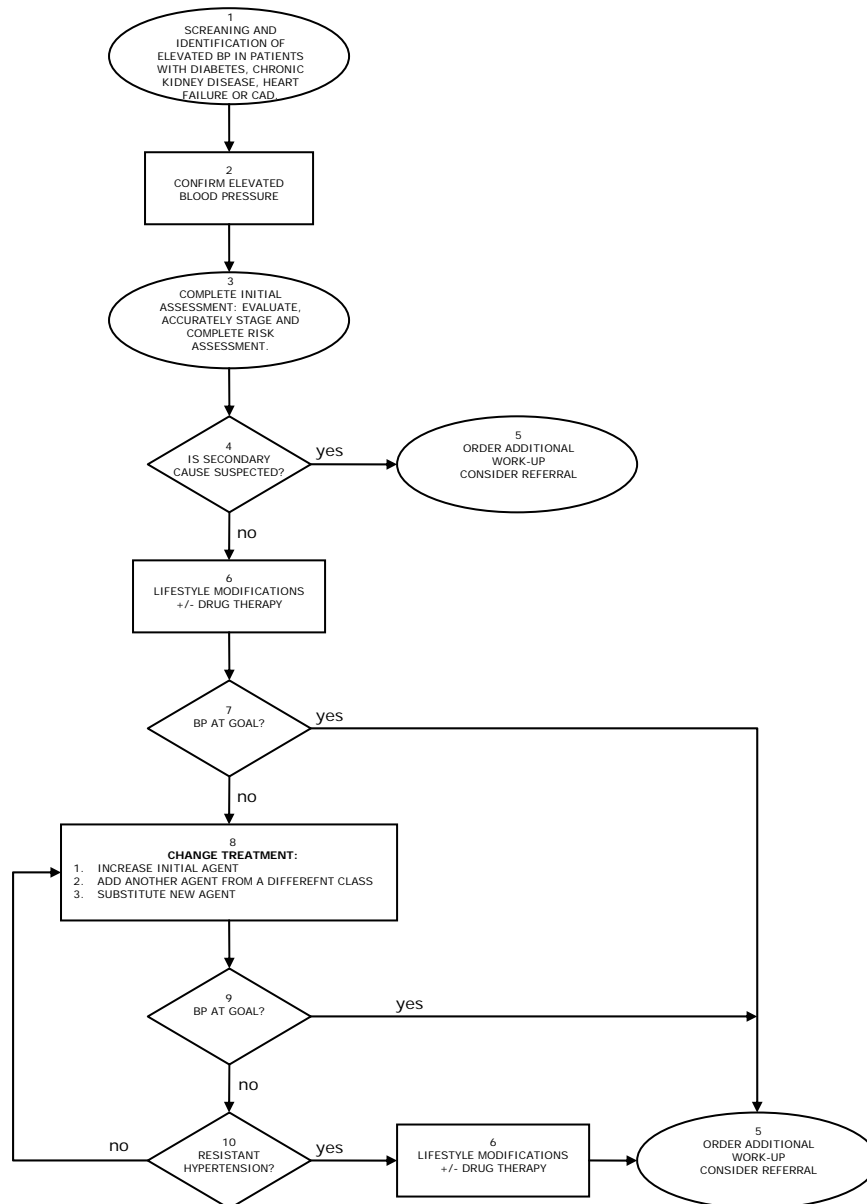


Figure 4. FIP on hypertension diagnosis and treatment.

In the next subsections the SDA* model is formally introduced, followed by the explanation of how sequences and cycles are made in the model, what non-determinisms the model is able to deal with, and the temporal model which is beneath the SDA* model.

3.1 Formal description

The SDA* model is introduced to represent knowledge on procedural activities in health care. In the next sections, the SDA* model is described in terms of the domain terms, the elements, and the connections that describe the health care procedure that is being formalized.

3.1.1 The Universe of Discourse²

Given D a particular disease, ailment, pathology, or syndrome, a finite set of **terms** $\mathcal{V}_D = \{v_1, \dots, v_n\}$ within the medical domain of D is defined to represent any descriptive or procedural health care knowledge on D . For example, the terms in the hypertension treatment contained in Figure 4 are seventeen: *screening-and-identification-of-elevated-BP*, *diabetes*, *chronic-kidney-disease*, *heart-failure*, *CAD*, *confirm-elevated-blood-pressure*, *complete-initial-assessment*, *secondary-cause-suspected*, *additional-work-up*, *consider-referral*, *life-style-modifications*, *drug-therapy*, *BP-at-goal*, *change-treatment*, *resistant-HT*, *HT-consult*, and *HT-continuing-care*.

Some of these terms are defined as **state terms** (i.e. $S_D \subseteq \mathcal{V}_D$ is the set of state terms). State terms represent facts that are useful to determine the condition of the patient in the process the SDA* model is describing. In the SDA* model, a **patient condition** contains all the terms observed for the patient in a particular moment (i.e. signs, symptoms, antecedents, taking drugs, secondary diseases, test results, etc.), therefore it is a subset of the set of terms \mathcal{V}_D .

The set of **decision terms** $\mathcal{D}_D \subseteq \mathcal{V}_D$ is the set of all the terms in \mathcal{V}_D that may be required by medical experts to choose among alternative medical, surgical, clinical, or management actions within the treatment of the disease D that the procedure globally describes. State and decision terms may be used to define any patient condition possible in D .

The set of **action terms** $\mathcal{A}_D \subseteq \mathcal{V}_D$ is the set of all the terms that represent the medical, surgical, clinical, or management actions that a doctor may decide on a patient in the course of the treatment of that patient's disease or health care procedure.

Though these three sets are not necessarily mutually disjoint, they together must contain all the feasible terms in D , i.e. $\mathcal{V}_D = S_D \cup \mathcal{D}_D \cup \mathcal{A}_D$. For example, in the above mentioned case of hypertension, $S_{hypertension} = \{\textit{screening-and-identification-of-elevated-BP}, *diabetes*, *chronic-kidney-disease*, *heart-failure*, *CAD*, *complete-initial-assessment*, *secondary-cause-suspected*, *BP-at-goal*, *HT-continuing-care*\}$, $\mathcal{D}_{hypertension} = \{\textit{secondary-cause-suspected}, *BP-at-goal*, *resistant-HT*\}$, and $\mathcal{A}_{hypertension} = \{\textit{confirm-elevated-blood-pressure}, *additional-work-up*, *consider-referral*, *life-style-modifications*, *drug-therapy*, *change-treatment*, *HT-consult*\}$ would be the set of state variables, decision variables, and action terms, respectively. Observe that the underlined terms are state and decision terms simultaneously.

3.1.2 Elements

The set of terms \mathcal{V}_D is used to define the three basic elements of the SDA* model: states, decisions, and actions. Formally speaking, a **state** s is a subset of state terms (i.e. $s \in \wp(S_D)$); a **decision** d is based on a subset D of decision terms (i.e. $D \in \wp(\mathcal{D}_D)$) and it is defined as a finite list

² In the first version of the SDA* model the universe of discourse is based on a set of the primitive medical terms that may be used to construct states, decisions, and actions. In forthcoming versions of the SDA* model, the universe of discourse will be extended to include *Boolean variables* (i.e. variables that are allowed to have two values: TRUE or FALSE), and later *multi-valued variables* (i.e. variables that can take one out of several possible values). This means that in this first version the health condition of a patient is defined exclusively by all the signs and symptoms this patient has.

$\langle D; D_1, D_2, \dots, D_k \rangle$, such that $D_i \in \wp(D)$ is a decision alternative (or **branch**), $D = D_1 \cup D_2 \cup \dots \cup D_k$, and $k \geq 0$ is the **branching factor** of the decision. An **action** a is a subset of action terms (i.e. $a \in \wp(\mathcal{A}_D)$).

From the point of view of semantics, a *state* (or *SDA* entry point*) describes an abstract patient condition in which all the terms in the state hold. For example, the state $\{\text{diabetes}, \text{complete-initial-assessment}\}$ represents all the patients with both diabetes and a complete initial assessment, but which may also have other possible features. From a logical point of view, a state is a conjunction of state terms. From a functional point of view, the states of a SDA* procedure are the entry points to that procedure or, in other words, the points where the treatment described can start.

If $C \subseteq (S_D \cup \mathcal{D}_D)$ is the current condition of a patient, we say a state s of a SDA* procedure is a **feasible entry point** of that patient in that procedure if and only if $s \subseteq C$. It may happen that one patient has several feasible entry points for the same SDA* under the same condition. It may also happen that one or several states are included in other states of the same SDA* (e.g. $s_1 \subseteq s_2$). In this case, every time s_1 is a feasible entry point, s_2 is also a feasible entry point. Empty states are also possible and they represent states in the SDA* procedure that any patient meets.

Observe that a state s that does not contain a state term v will be a feasible entry point to both patients whose condition comprises v and patients whose condition does not comprise v (see Table 1). If we want to change this behavior we have to define two terms for the same health care concept, one being the negation of the other one (e.g. *diabetes* and *not-diabetes*). This way, a state containing the term *not-diabetes* (i.e. negation of *diabetes*) will not be a feasible entry point for diabetic patients whose condition does not comprise *not-diabetes*.

	$v \in \text{PATIENT CONDITION}$	$v \notin \text{PATIENT CONDITION}$
$v \in s$	s is a feasible entry point	s is NOT a feasible entry point
$v \notin s$	s is a feasible entry point	s is a feasible entry point

Table 1. Basic logic rule of feasible entry points.

A *decision* (or *SDA* branching point*) describes a point of the SDA* where the treatment can follow alternative courses of action depending on which are the decision terms the treated patient meets. For example, the set of decision terms $D = \{\text{stage1-HT}, \text{stage2-HT}, \text{low-BP}, \text{BP-at-goal}\}$ could be used to propose alternative treatments whether the patient is hypertensive ($D_1 = \{\text{stage1-HT}, \text{stage2-HT}\} \subseteq D$), hypotensive ($D_2 = \{\text{low-BP}\} \subseteq D$) or none ($D_3 = \{\text{BP-at-goal}\} \subseteq D - (D_1 \cup D_2)$). From a logical point of view, a decision represents a disjunction of conjunctions on a set of decision terms. From a functional point of view, decisions allow the represented SDA* to be as general and flexible as to combine several variations on the treatment of a disease, and to make the application of these variations depend on the particularities of the patient.

If $C \subseteq (S_D \cup \mathcal{D}_D)$ is the current condition of a patient and $d = \langle D; D_1, D_2, \dots, D_k \rangle$ a decision element of a SDA* procedure, we say D_i is a **feasible branch** for that patient if and only if $D_i \subseteq C$. One or several branches may contain none decision variable; in this case, all these branches are feasible. It may also happen that in the same decision two or more branches totally or partially overlap. In the first case (i.e. $D_i \subseteq D_j$), D_i will be a feasible branch whenever D_j is feasible, and D_j

will not be a feasible branch if D_i is not. In the second case (i.e. $(D_i \cap D_j) \neq \emptyset$) each situation must be studied separately. Observe that if $D_i = D_j$, both branches are evaluated the same for any possible patient. Empty conditions are always feasible.

Concerning the branching factor k of a decision, it must be zero, one, or greater than one. An SDA* decision with a branching factor of zero or one is interpreted as unfinished element, maybe because at the time of developing of the SDA* there is not health evidence on how to branch patients at that point of care. A branching factor $k=0$ transforms a SDA* decision into a dead end element. A branching factor $k=1$ acts as a filter of the patients that may proceed with the treatment at the decision point. A branching factor $k=2$, allows the construction of SDA* **binary decisions** as $\langle D; D_1, D-D_1 \rangle$. Observe that binary decisions are not equivalent to IF-THEN-ELSE structures since any patient with a condition C containing all the decision terms in D will make both branches of the above decision feasible. Like in the case of the state elements, IF-THEN-ELSE behaviors may be achieved through the definition of contrary terms (e.g. *diabetes* and *not-diabetes*), and the definition of decisions as $\langle \{diabetes, not-diabetes\}; \{diabetes\}, \{not-diabetes\} \rangle$. In this case, diabetic patients will follow the first branch, and non diabetic patients the second one.

An alternative interpretation of a decision $d = \langle D; D_1, D_2, \dots, D_k \rangle$ is that it is based on a “fictitious” variable d whose domain (i.e. the values that the variable can take) is D , and each branch D_i is a subset of these possible values. For example, $BP = \langle \{stage1-HT, stage2-HT, low-BP, BP-at-goal\}; \{stage1-HT, stage2-HT\}, \{low-BP\}, \{BP-at-goal\} \rangle$.

Let us observe that a branch $D_i = \emptyset$ of a decision d is always feasible for any patient arriving to d . If a patient condition C includes none of the branches of a SDA* decision $\langle D; D_1, D_2, \dots, D_k \rangle$ (i.e. $D_i \not\subset C$ for all $i=1, 2, \dots, k$), then none of the branches is feasible, and the decision becomes a dead end element of the SDA* procedure for all the patients under that condition. In order to avoid this situation a SDA* decision can contain an **otherwise branch** (i.e. $\langle D; D_1, \dots, D_k, otherwise \rangle$) which is feasible only if the patient condition C makes none of other branches feasible. For example, $BP = \langle \{BP-at-goal\}; \{BP-at-goal\}, otherwise \rangle$ that represent the decisions #7 and #9 in the FIP of Figure 4.

An *action* element (or *SDA* action block*) describes a group of actions in the SDA* procedure. These elements do only represent action proposals whose application must be seen out of the SDA* model. So, if the SDA* suggests the physician to prescribe a beta-blocker it is up to the physician to decide whether the drug is finally prescribed or not, and it is up to the patient (or some other person) to make sure that the patient takes the drug. This means that two sequential actions in the SDA* model do not necessarily represent a sequential execution of the actions in the real world, but consecutive action proposals within the SDA* procedure.

The SDA* model does not distinguish between instant actions (i.e. those actions with an immediate end as for example an expert recommendation) and abiding actions (i.e. those actions which extend in time as for example starting an assessment process that may last several days). The reason is that actions in the SDA* model represent the launch of the action, regardless whether this is an instant or an abiding action in the real world. Typical sorts of actions are: recommendations (e.g. stop-smoking, start-soft-exercise, avoid-salt-in-meals, etc.); prescriptions; radiographies; analyses; medical, surgical or clinical procedures; specialist consultations;

application of an alternative SDA* procedures, etc. From a functional point of view, action blocks represent the core elements of the SDA* model since the final purpose of this model is to represent health care procedures as a combination of actions.

Each action term in an action element has two constraints: the first one (called the **set of petitioners**) is on the sort of actors that are allowed to request the action (e.g. only medical doctors are allowed to prescribe drugs). The second one (called the **set of performers**) is on the sort of actors that are allowed to perform the action in the real world (e.g. injecting some drugs can be restricted to nurses and to medical doctors, but some other drugs can also be injected by the own patient or some relative). These constraints on the actions permit the description of collaborative medical treatments in which several professionals may interact. Any petitioner in the set of petitioners is allowed to requests the action to be executed. Any performer in the set of performers is allowed to execute the action.

Action blocks are independent of the patient condition; therefore they use to be preceded either by a state that describes what the state of a patient should be in order to deserve that action, or by a decision that determines whether the patient meets the features required for the action to be applied. Empty action blocks have the meaning of “do nothing”, which is the same as not having the action block in the SDA*.

Flowcharts are used to represent SDA* procedures in a graphical way. Figure 5 shows how states, decisions, and actions are represented in this sort of flowcharts.



Figure 5. Elements of the SDA* Model.

The correct combination of states, decisions, and actions allows the construction of explicit health care procedural knowledge within the SDA* model. This combination of elements is made by means of connectors.

3.1.3 Connectors

This section explains how the SDA* elements introduced in the previous section can be combined to form proper health care procedures.

In the SDA* model, a **connector** is defined as an arrow that goes from one element in the input of the connector (or in-element) to another element in the output of the connector (or out-element).

From the point of view of the SDA* elements, any state is an in-element of one connector³, but it may be an out-element of any number of connectors in the FIP (including none). Decisions are in-elements of as many connectors as the branching factor of the decision, and out-elements of one or several connectors in the SDA* procedure. Finally, actions are in-elements of one only

³ This constraint will be relaxed with the introduction of type-2 non-determinism in section 3.3.

connector, and out-elements of at least one connector. These restrictions are graphically shown in Figure 6.

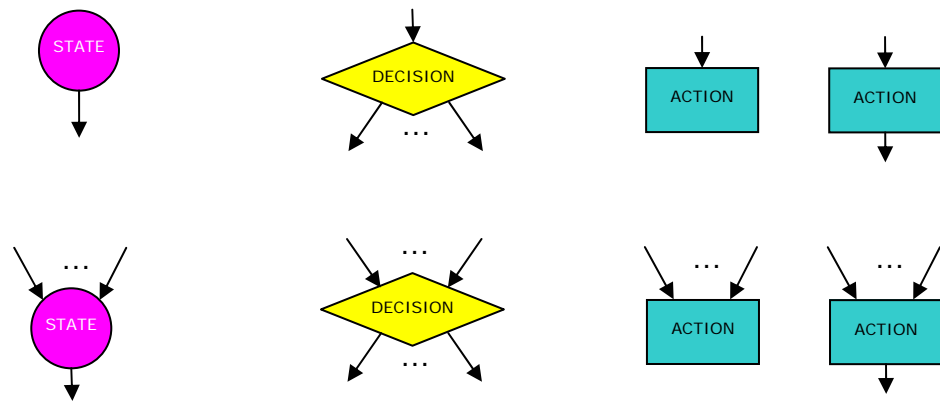


Figure 6. Feasible element connections in the SDA* Model.

In that figure, the up-left state and the bottom-left state describe a situation in which all the patients whose condition makes the state a feasible entry point evolve following the outgoing connector. The difference between them is that in the first case the SDA* procedure do not inform about when a patient can reach that state in the middle of a treatment (i.e. it is an input state of the health care procedure). In the second case, the state can be either an input state of the health care procedure for new incoming patients, or an intermediate state which is reached after the application of any of the elements in the incoming connections of the state.

A decision was defined as a list $\langle D; D_1, D_2, \dots, D_k \rangle$ of sets of decision terms; D being all the possible terms in the decision, D_i a subset of D for all $i=1..k$, and k the branching factor. Each alternative D_i in the decision is assigned a different outgoing connector of the decision. The meaning of a decision point is that any patient reaching the decision (by one of the incoming connectors) may follow any of the outgoing connectors whose D_i is contained in the patient condition (i.e. one of the feasible branches of the decision).

An action block contains all the action terms that are to be suggested to deal with the patient reaching that element. The up-right actions in Figure 6 describe types of action blocks that are only reachable from one element in the SDA*. Within this group, the left one describes a terminal action in which the information of how to proceed after the action is not provided by the health care procedure. The actions in the bottom are general cases describing action blocks to follow after the application of any of the elements in the action incoming connectors. They also act as a joint of several courses of action of the SDA* procedure that converge to stop (action on the left) or that converge into one single action block to propose the same group of actions and then proceed in the same way through the action block outgoing connector (action on the right).

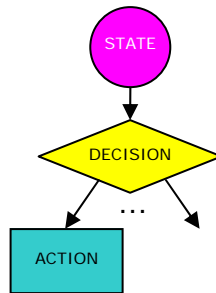


Figure 7. SDA sequence.

3.2 Sequences and cycles

The basic structure of the SDA* model is **the SDA sequence** that connects one state with a decision and each branch of that decision with an action. Figure 7 represents this basic structure.

The SDA sequence can be simplified with the elimination of one or several of the elements in the sequence. So, the elimination of the state must be interpreted as if there is not a health care reason to describe the state of the patient at this point of care (e.g. lack of medical meaning, medical irrelevance, cause of confusion, disagreement, etc.). Sometimes, the application of a set of actions is mandatory for all the patients arriving to the SDA sequence. In this case the decision element is eliminated and only one action block with all the common actions is connected after the state. Sometimes, a decision element is not enough to arrive to a conclusion about the sort of actions to carry on or the representation of all the possibilities with a single decision is confusing. In these cases the action block must be eliminated from the SDA sequence in order to chain several decisions. All these cases of **SDA sequence reduction** are depicted at the top of Figure 8.

At the bottom of Figure 8 the cases of elimination of two elements of a SDA sequence are represented. The left side case describes a situation in which two (or more) states from consecutive SDA sequences are connected. Although this is a correct sequence, there is not a clear reason that justifies it since a sequence of states is equivalent to a single state containing the state terms of all the states in the sequence. The case in the middle represents a sequence that connects two decisions. This is a common practice in the construction of health care procedures with the SDA* model. The last case in the bottom-right side represents a sequence of two (or more) actions of consecutive SDA sequences directly connected. Like it happened with the states in the first case, this sequence is better replaced by a single action containing all the action terms of the action blocks involved in the sequence.

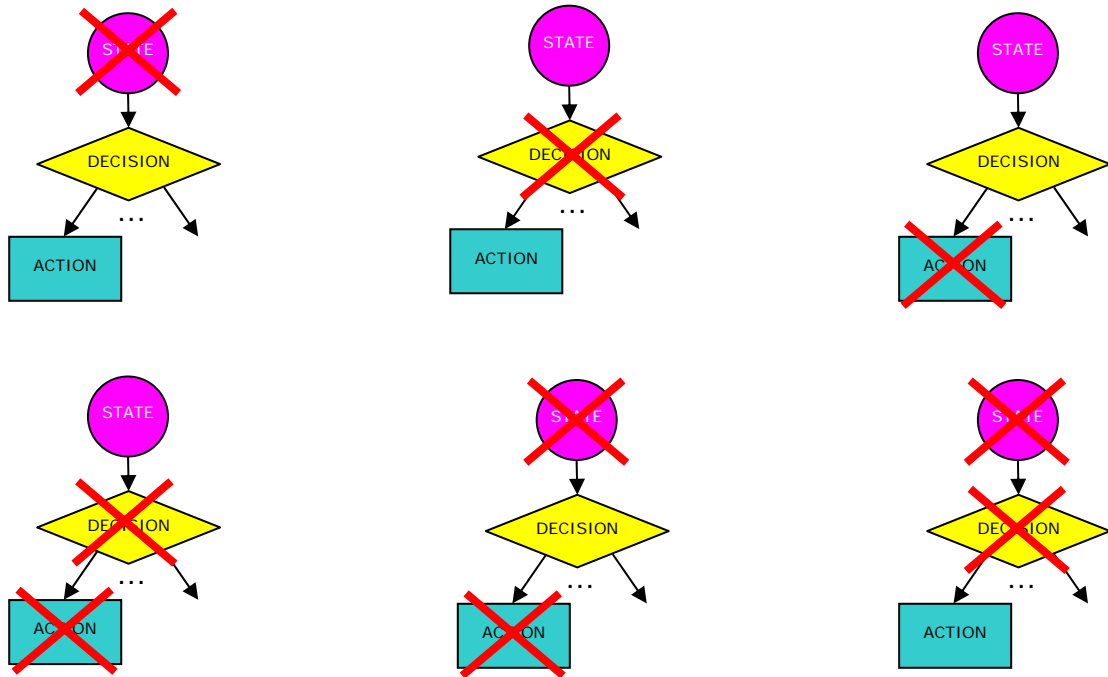


Figure 8. Simplified SDA sequences.

SDA sequences (and their reductions) can be concatenated by means of connectors. Figure 9 shows the most general case of a **SDA sequence concatenation** where none of the elements in the SDA sequences have been eliminated.

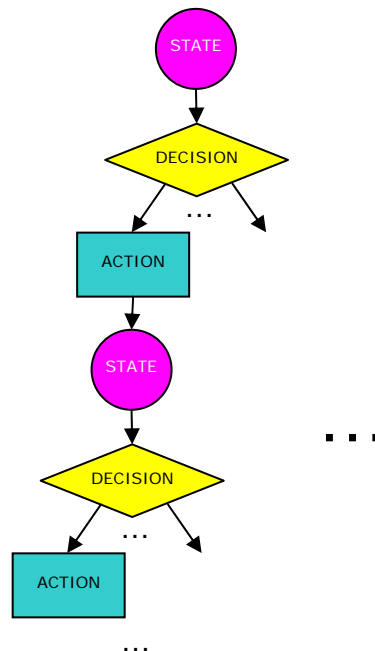


Figure 9. Concatenation of SDA sequences.

Apart of sequences, the SDA* model can represent cycles. A **cycle** is defined as a repeated sequence of elements in a SDA* procedure. Cycles may be used to represent repetitions in a medical process or jumps to an already previously observed situation in the course of action

followed. Cycles in this model do not have explicit termination conditions; the exit of a cycle occurs when one of the decisions of the cycle drives the patient to an outgoing connection which is not part of the cycle.

3.3 Non-Determinism

Determinism is the principle by which every event, act, and decision (effect) is the consequence of some antecedents (causes). In healthcare, these causes can be medical, surgical, genetic, environmental, managerial, familiar, social, etc. On the contrary, **non-determinism** states that there are events which do not correspond to a cause. Historically, there have been defined three **types of non-determinisms**: one that holds that some events are uncaused (e.g. from a practical point of view, in healthcare, uncaused events are equivalent to events with an unknown unfindable cause), another one that holds that there are nondeterministically caused events (e.g. a physician that follows alternative therapies for equivalent cases without an explicit explanation), and the third one that holds that there are agent-caused events (e.g. external events like the arrival of a patient whose health condition allows the treatment to start at different points). The SDA* model can deal with all the above types of non-determinism. As a consequence of this, for the same situation (i.e. patient condition) a non-deterministic SDA* is able to represent several different interventions with no support to decide which one should be followed. A non-deterministic SDA* procedure may propose more than one intervention and it must be the physician the final responsible of the selection.

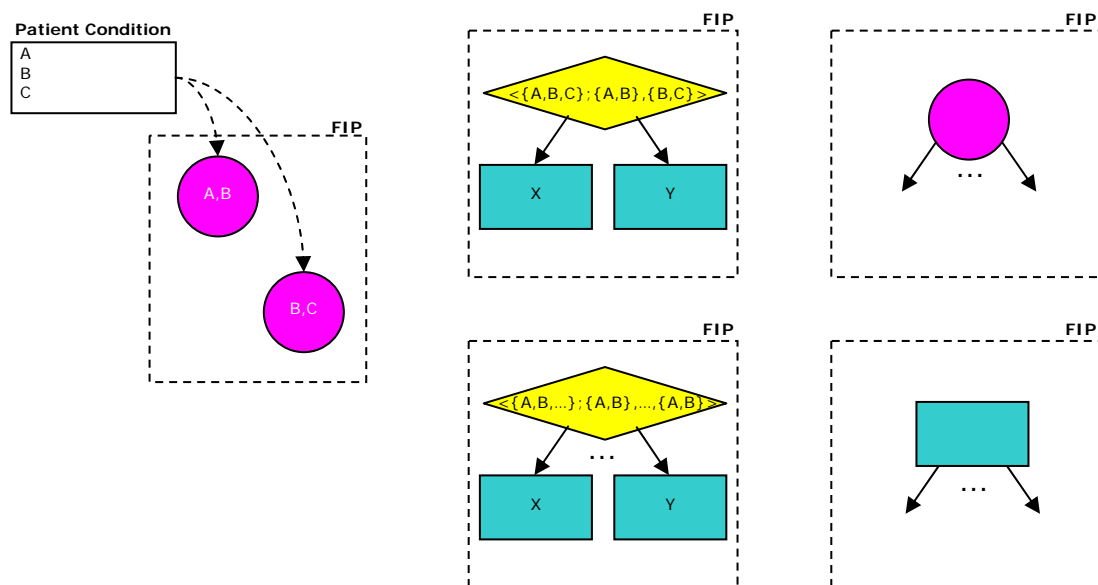


Figure 10. Non-Determinism in the SDA* Model.

Figure 10 shows the three sorts of **non-determinism in the SDA* model** that can be observed in a SDA* FIP. From left to right, the first case (**type-0 non-determinism**) describes the situation in which a patient with a particular condition can match several states at the same time and therefore be non-deterministically recommended to start one out of several alternative

interventions. In the second case (**type-1 non-determinism**), the current condition of a patient can satisfy several branches of the same decision, and therefore be able to follow any of them. For example, if the patient condition is $\{high\text{-}blood\text{-}pressure, taking\text{-}drugs\}$, then any branch of the sort $\{\}$, $\{high\text{-}blood\text{-}pressure\}$, $\{taking\text{-}drugs\}$, or $\{high\text{-}blood\text{-}pressure, taking\text{-}drugs\}$ is a feasible branch. The last case in the right side of Figure 10 (**type-2 non-determinism**) describes a situation in which either a state or an action in the FIP are in-elements of several connectors. Here, the SDA* procedure introduces two or more alternative paths that patients going out of these elements may (or may not) non-deterministically follow.

3.4 Time

The time model establishes two sorts of temporal constraints: those which are related to the terms in a SDA* element and those others related to the connectors. Each term and connector may optionally have one constraint or not. The time constraints of the terms are of the sort **[start, end, frequency]** and they mean that the term is observed from the start time, to the end time with the frequency indicated. For example, when $v = (antidepressant, [3w, 1d, 24h])$ is a state term it means that the state of the patient is conditioned by the fact that “(with respect to the current moment) he has been taking one antidepressant every day since three weeks ago to one day ago”. Observe that taking two antidepressant units should be said $(twoAntidepressant, [x, y, 24h])$ or $(antidepressant, [x, y, 12h])$ if the units are taken together or in two doses, respectively. The first case can also be represented by introducing the term v in the state two times.

If v is a decision term the meaning is equivalent to the question “has the patient been daily taking one antidepressant between three weeks ago and yesterday?”. But if it is an action term the meaning is an order of taking that antidepressant starting in the start time and ending in the end time with the frequency indicated in the frequency value (i.e. a prescription). In this case, start must be a nearer to the current time than end.

s	seconds
m	minutes
h	hours
d	days
w	weeks
M	months
y	years

Table 2. Time units in the SDA* model.

In the SDA* model, this sort of terms with a time constraint are called **temporal terms**.

The second sort of time constraints in the SDA* model is related to the SDA* connectors and it has the form **[min, max]**. They are optional and represent delays (or durations). Both values are also optional. A connector with such time constraint indicates that the evolution from the in-element to the out-element of the connector takes between *min* and *max* times. If only the *min* value is present, it means that the connector can be crossed only if a *min* interval of time passes. If only the

max value is in the constraint, the meaning is that the connector can be followed not later than a *max* interval of time.

The sort of temporal units of the *start*, *end*, *frequency*, *min*, and *max* components of the time constraints are the ones included in Table 2, and any of these values is represented by a natural number followed by one of these temporal units⁴. For example, 15s for “fifteen seconds”, 5m for “five minutes”, 3h for “three hours”, 4d for “four days”, 7w for “seven weeks”, 10M for “ten months”, and 3y for “three years”.

States and decisions describe past or current aspects of the patient, and therefore the temporal constraint of *start* must be bigger than the temporal constraint of the *end* (e.g. [3d, 2d] or [1y, 3w]). On the contrary, action elements represent future actions and the start value of a temporal constraint must be smaller than the temporal constraint of the end (e.g. [1d, 3y] or [1h, 6d]).

3.5 Parallelism

Parallelism is admitted by the SDA* model but in a ***patient-oriented*** (instead of a procedure-oriented) fashion. From the point of view of the patient following a SDA* procedure, this person has a single treatment in which several events may concur in time. In this approach parallelism does not mean that the patient is following several treatments at the same time (this will be a procedure-oriented approach), but that the actions of the treatment overlap in time.

This idea must be conceived together with the fact that SDA* procedures do not represent the health care procedures themselves, but the indications of what health actions have to be started now and, expectedly, in the future. Parallel to the SDA* procedure, the evolution of the real patient in the real world is what conditions how to apply the SDA* procedure in the next encounter with the patient. In other words, the SDA* procedure suggests a set of actions according to the current patient condition, and provides a farther perspective of how the treatment of this patient should be in the future, based exclusively on the limited current evidence provided by the current state of the patient and not on the real future evolution of the patient. Of course, this perspective is founded on both health care knowledge and experiences about the feasible evolutions of patients in the disease the SDA* procedure is dealing with.

In this context, all the action terms of an action block are launched in parallel, subject to their respective temporal constraints. In Figure 11 the action terms A_i and A_j , belonging to the same action block, have a parallel region where both behave simultaneously on the patient. These actions may also be in parallel to actions as A_k from other action blocks, as the figure also depicts.

⁴ Other temporal concepts as “**now**”, “**birth-time**”, and “**death-time**” are also possible.

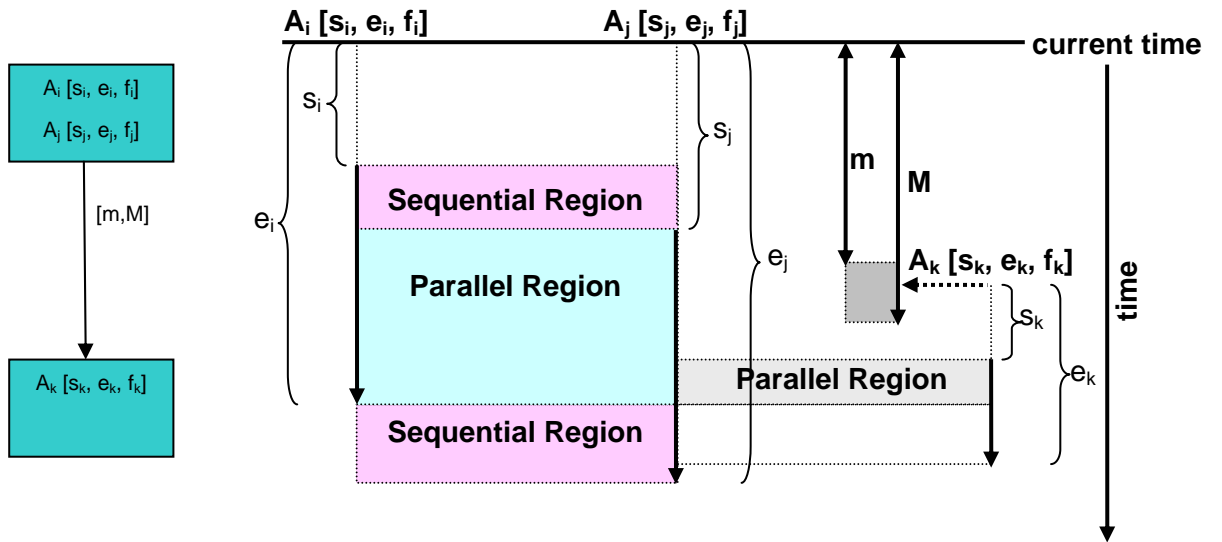


Figure 11. Parallel actions in time.

4 Construction and execution of health procedures with the SDA* Model

This section introduces the procedures that the SDA* model is able to describe as an abstract data type (ADT), providing the specification of a basic functional interface to manage the construction of such health procedures. An XML Schema is proposed that provides the structure to represent SDA* procedures as XML documents. The ADT functions are used to describe the execution of a health procedure under the SDA* model. The section finishes with several examples of SDA* procedures in the K4CARE project.

4.1 Abstract data type SDA* procedure

This section aims at providing a formal proposal about the **basic constructors** that any system capable of defining SDA* procedures is recommended to have. This proposal follows the definitional notation of formal specification of abstract data types [5].

<i>time & actors</i>	$\text{TIME} = \lambda \mid \text{NUMBER}\{s \mid m \mid h \mid d \mid w \mid M \mid y\}$ $\text{PETITIONERS} = \text{Set of ACTOR}$ $\text{PERFORMERS} = \text{Set of ACTOR}$
<i>elements</i>	$\text{EmptyState}: \rightarrow \text{STATE}$ $\text{InsertTerm}: \text{Term} \times [\text{TIME}]^3 \times \text{STATE} \rightarrow \text{STATE}$ $\text{EmptyBranch}: \rightarrow \text{BRANCH}$ $\text{OtherwiseBranch}: \rightarrow \text{BRANCH}$ $\text{InsertTerm}: \text{Term} \times [\text{TIME}]^3 \times \text{BRANCH} \rightarrow \text{BRANCH}$ $\text{EmptyDecision}: \rightarrow \text{DECISION}$ $\text{InsertBranch}: \text{BRANCH} \times \text{DECISION} \rightarrow \text{DECISION}$ $\text{EmptyAction}: \rightarrow \text{ACTION}$ $\text{InsertTerm}: \text{Term} \times [\text{TIME}]^3 \times \text{PETITIONERS} \times \text{PERFORMERS} \times \text{ACTION} \rightarrow \text{ACTION}$
<i>SDA*</i>	$\text{EmptySDA*}: \rightarrow \text{SDA*}$ $\text{InsetElement}: \text{Element} \times \text{SDA*} \rightarrow \text{SDA*}$ $\text{InsertConnector}: \{\text{STATE} \mid \text{ACTION}\}^2 \times [\text{TIME}]^2 \times \text{Element} \times \text{SDA*} \rightarrow \text{SDA*}$ $\text{InsertConnector}: \text{BRANCH} \times \text{DECISION} \times [\text{TIME}]^2 \times \text{Element} \times \text{SDA*} \rightarrow \text{SDA*}$

Table 3. SDA* Abstract data type: basic constructors.

Patient states, branches, and actions are sets containing temporal terms of the form (term, [time₁], [time₂], [time₃])⁵, any of the three times being optional; *decisions* are sets of *branches*, and *SDA** are sets that contain either *states*, *decisions*, *actions*, or *connectors*, where connectors can be elements of the form (sa₁, sa₂, [time₁], [time₂]) or (branch, decision, [time₁], [time₂]), sa_i standing for a *state* or an *action*.

⁵ Otherwise branches are an exception. They are branches without terms.

4.2 Textual representation of SDA* procedures

The procedures of the SDA* model can be expressed in textual format. Table 4 shows the body of the XML Schema to define SDA* procedures as XML files.

```

...
<xs:simpleType name="sda_time">
  <xs:restriction base="xs:string">
    <xs:pattern value="[0-9]*[smhdwMy]" />
  </xs:restriction>
</xs:simpleType>
<xs:complexType name="sda_term">
  <xs:sequence>
    <xs:element name="start" type="sda_time" minOccurs="0" />
    <xs:element name="end" type="sda_time" minOccurs="0" />
    <xs:element name="frequency" type="sda_time" minOccurs="0" />
  </xs:sequence>
  <xs:attribute name="name" type="xs:string" use="required" />
</xs:complexType>
<xs:complexType name="sda_actionterm">
  <xs:sequence>
    <xs:element name="start" type="sda_time" minOccurs="0" />
    <xs:element name="end" type="sda_time" minOccurs="0" />
    <xs:element name="frequency" type="sda_time" minOccurs="0" />
    <xs:element name="petitioner" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
    <xs:element name="performer" type="xs:string" minOccurs="0" maxOccurs="unbounded" />
  </xs:sequence>
  <xs:attribute name="name" type="xs:string" use="required" />
  <xs:attribute name="type" type="xs:string" />
</xs:complexType>
<xs:complexType name="sda_connector">
  <xs:sequence>
    <xs:element name="min" type="xs:time" minOccurs="0" />
    <xs:element name="max" type="xs:time" minOccurs="0" />
    <xs:element name="element" type="xs:string" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="sda_branch">
  <xs:sequence>
    <xs:element name="sda_term" type="sda_term" minOccurs="0" maxOccurs="unbounded" />
    <xs:element name="sda_connector" type="sda_connector" />
  </xs:sequence>
</xs:complexType>
<xs:complexType name="sda_state">
  <xs:sequence>
    <xs:element name="sda_term" type="sda_term" minOccurs="0" maxOccurs="unbounded" />
    <xs:element name="next" type="sda_connector" minOccurs="0" />
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="required" />
</xs:complexType>
<xs:complexType name="sda_decision">
  <xs:sequence>
    <xs:element name="sda_branch" type="sda_branch" minOccurs="0" maxOccurs="unbounded" />
    <xs:element name="otherwise" type="sda_connector" minOccurs="0" />
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="required" />
</xs:complexType>
<xs:complexType name="sda_actionblock">
  <xs:sequence>
    <xs:element name="sda_action" type="sda_actionterm" minOccurs="0" maxOccurs="unbounded" />
    <xs:element name="next" type="sda_connector" minOccurs="0" />
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="required" />
</xs:complexType>
<xs:complexType name="sda_procedure">
  <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:element name="sda_state" type="sda_state" />
    <xs:element name="sda_decision" type="sda_decision" />
    <xs:element name="sda_action" type="sda_actionblock" />
  </xs:choice>
</xs:complexType>
...

```

Table 4. SDA* XML Schema.

4.3 Execution of SDA* procedures

One of the key aspect to fully understand procedures described under the SDA* model is to know how they are executed for a particular patient. Before going on, we must recall that these procedures are formal representations of the general intervention to deal with a particular disease, ailment, pathology or syndrome. They are not representations of the evolutions of the patients under such circumstances. This means that applying the indications of a SDA* to a particular patient does not necessarily imply that this patient will evolve the way the SDA* indicates. This fact describes a parallel view of the problem at **two levels**, the level of the course of actions indicated in the SDA* (medical knowledge) and the level of the evolution of the patients that follow the SDA* (reality or medical data).

This duality may disturb or confuse the reader. However, this same reader must think of the SDA* as indications that a particular patient may follow, may not follow, follow in part, or follow during not enough time; or, also common in medicine, even strictly following the indications, the patient may evolve unexpectedly.

The practice of medicine is universally based on the encounters between the patients and the healthcare professionals. In a particular **encounter** a patient exhibits a specific **condition** within the disease he is assisted for. The role of the health care professional in the encounter is to interpret these signs, symptoms, and the rest of the information provided in order to conclude about the set of **actions** to follow (e.g. recommendation, prescription, procedure, etc.).

In the SDA* model, a **patient condition** is described as a set of temporal terms representing the patient current condition, including the patient health antecedents. These terms can be state, decision or action terms. For example, {(ElevatedBloodPressure), (BPAtGoal, [t₁,t₂,-]), (Antidepressant, [t₃, t₄, t₅])} is the condition of a patient that has an elevated blood pressure (i.e. BP_≥140/90), but who has had the pressure at goal between the times t₁ and t₂, and who has been taking antidepressant between t₃ and t₄ with a frequency t₅.

<i>patient</i>	EmptyCondition: → PCONDITION
<i>condition</i>	InsertTerm: Term × [TIME] ³ × PCONDITION → PCONDITION

Table 5. Patient condition abstract data type: basic constructors.

Given a patient condition and a SDA*, both based on the same set of terms, the execution of that SDA*procedure for that patient starts at any of the states of the SDA* that are feasible entry points. If it is required, the health care professionals in the encounter can select a subset of all the alternatives. Each feasible entry point in the selected set starts alternative feasible treatments of the patient.

A treatment consists of all the action (temporal) terms found in a SDA* **path that starts in a feasible entry point and finishes either in a state that is not a feasible entry point (i.e. the patient condition must change before evolving in this line) or in a connector with a temporal range [min, max] with min>0 (i.e. the SDA* procedure sets a temporal break before the patient treatment can continue)**. This path is a sequence of SDA* elements, each one being an out-element of a connector with in-element the pervious element in the sequence. If the in-element is a decision, all the branches that the patient condition meets (or the branch otherwise if none

meets), i.e. the feasible branch of the decision, can be followed. If an element is non-deterministically connected to other elements all the non-deterministic connectors can be followed. In both cases, it is the health care professional who selects the treatment to apply among all the feasible alternatives supported by the SDA* procedure.

We define a temporal term $(\forall, [t_1, t_2, t_3])$ is **equally or more restrictive** than another temporal term $(\forall, [t_1', t_2', t_3'])$ if the following conditions hold:

1. $(t_1' \leq t_1)$ or $(t_1'$ is void).
2. $(t_2 \leq t_2')$ or $(t_2'$ is void) or $(t_2' = 0)$.
3. $(t_3 \leq t_3')$ or $(t_3$ is void)

Given a patient condition, we say a SDA* state is a feasible state if all the terms in the state can be found in the patient condition and they are equally or more restrictive in the state than in the patient condition. In a similar way, we say a branch of a SDA* decision is feasible if all the terms in the branch are in the patient condition contains and they are equally or more restrictive in the branch than in the patient condition.

For example, a SDA* state $\{[beta-blocker, 1M, 1w, 1d]\}$ (i.e. patients taking one beta-blocker per day during the last month until last week) will be feasible for patients that meet condition c1 in Table 6 (more restrictive), but not feasible for patients meeting conditions c2 (the patient has been taking the drug since more time than one month ago), c3 (the patient has been taking beta-blockers during the last week, just in contradiction with the indication of stop taking beta-blockers one week ago pointed out by the state), or c4 (not only the frequency but the duration of medication is shorter).

Provided a decision term $[highBloodPressure, 1M, 3d, -]$ (i.e. true if the patient has got high blood pressure since one month ago till recently), all the patients meeting conditions c5 or c6 in Table 6 (more restrictive) will evaluate the decision term to true, but not c7.

Condition	Expression	Meaning
c1	$[beta-blocker, 1y, 1d, 12h]$	<i>Taking a beta-blocker every twelve hours since one year ago till yesterday.</i>
c2	$[beta-blocker, -, 2d, 8h]$	<i>Taking beta-blocker every eight hours till two days ago (since much time ago).</i>
c3	$[beta-blocker, 1w, -, 8h]$	<i>Taking beta-blocker every eight hours since one week ago (till now).</i>
c4	$[beta-blocker, -, -, 3d]$	<i>Taking beta-blocker every eight hours since much time ago till now.</i>
c5	$[highBloodPressure, 2M, 1d, -]$	<i>Till yesterday, Blood Pressure has been high during the last two months.</i>
c6	$[highBloodPressure, 2M, -, -]$	<i>Blood Pressure has been high during the last two months (and it is now).</i>
c7	$[highBloodPressure, 3w, -, -]$	<i>Blood Pressure has been high during the last three weeks (and it is now).</i>

Table 6. Some examples of temporal terms and meaning.

4.4 Examples

This section contains partial and complete examples of SDA* procedures. The action terms have been classified into recommendations, prescriptions, radiographies, analyses, procedures, specialists, FIPs, and any, as Table 7 summarizes.

[RECOMMENDATION]:	[REC]:	Variable that represents a recommendation of the physician.
[PRESCRIPTION]:	[PRES]:	Variable that represents a drug prescription.
[RADIOGRAPHY]:	[RAD]:	Variable that represents an order of radiography.
[ANALYSIS]:	[ANA]:	Variable that represents an order of analysis.
[PROCEDURE]:	[PROC]:	Variable that represents the application of a procedure.
[SPECIALIST]:	[SPEC]:	Variable that represents the specialist the patient is derived to.
[FIP]:	[FIP]:	Variable that represents the execution of another FIP.
[ANY]:	[ANY]:	Variable that represents any sort of action.

Table 7. Sorts of action variables.

The treatment of hypertension in 4.4.2 is taken from the guideline published by the Institute for Clinical Systems Improvement (www.icsi.org) in the National Guideline Clearinghouse in the USA. The rest of procedures were taken from the “Consensus Guidelines for Assessment and Management of Depression in the Elderly“ of the NSW Health Department in Australia. An exception is presented in subsection 4.4.3, where the K4CARE procedure [6] for Comprehensive Assessment of homecare patients is represented following the SDA* model.

4.4.1 Representing partial knowledge

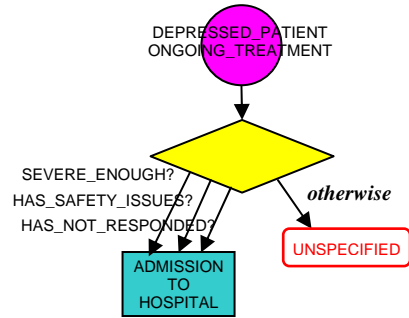
Many times clinical practice guidelines contain valuable knowledge that is apparently disconnected from other pieces of knowledge that may appear in the same guideline. This kind of knowledge is usually represented as text. This section contains some examples of textual knowledge pieces extracted from real guidelines and it shows how they could be represented in the SDA* model.

<p>“Medication is likely to be needed where there is any sustained depressive disorder and when non-pharmacological strategies are not achieving their goals”</p>	<p>STATE VARIABLES: SUSTAINED_DEPRESSIVE_DISORDER ONGOING_PHARMA_STRATEGY</p> <p>DECISION VARIABLES: SUCCESSFUL_PHAMA_STRATEGY?</p> <p>ACTION VARIABLES: MEDICATION</p> <pre> graph TD A((SUSTAINED_DEPRESSIVE_DISORDER ONGOING_PHARMA_STRATEGY)) --> B{SUCCESSFUL_PHAMA_STRATEGY?} B --> C[MEDICATION] B -- otherwise --> D[UNSPECIFIED] </pre>
<p>“Useful signs to indicate commencing medication are:</p> <ul style="list-style-type: none"> • Presence of biological signs, disturbed sleep, appetite and energy changes • Diurnal variation in mood • Agitation or retardation • Depression with any psychotic features.” 	<p>STATE VARIABLES: BIOLOGICAL_SIGNS DISTURBED_SLEEP DISTURBED_APPETITE ENERGY_CHANGES MOOD_VARIATION AGITATION_RETARDATION PSYCHOTIC_FEATURES DEPRESSION</p> <p>ACTION VARIABLES : START_MEDICATION</p> <pre> graph TD A((BIOLOGICAL_SIGNS DISTURBED_SLEEP DISTURBED_APPETITE ENERGY_CHANGES)) --> B[START_MEDICATION] C((MOOD_VARIATION)) --> B D((AGITATION_RETARDATION)) --> B E((PSYCHOTIC_FEATURES DEPRESSION)) --> B </pre>

“Admission to hospital can be essential where the depression:

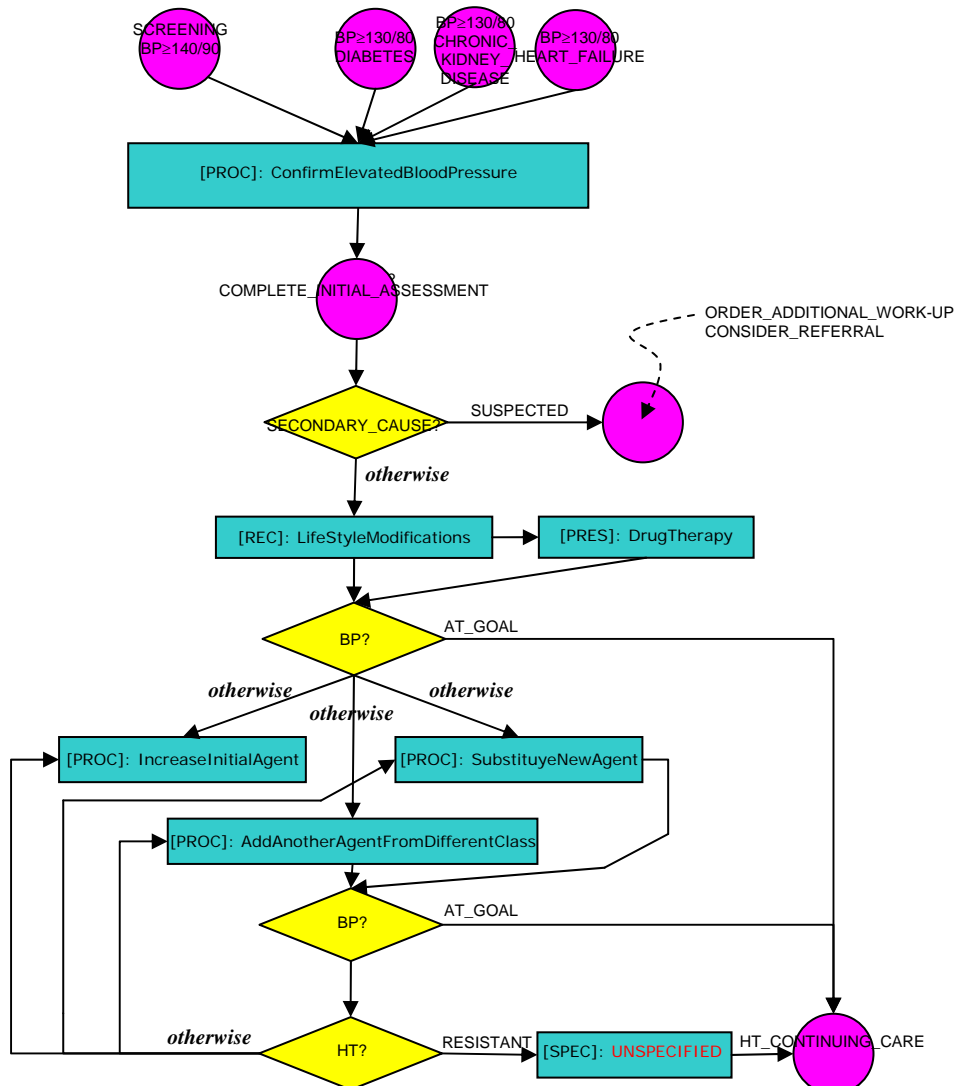
- Is severe enough to impair reasonable daily living function and supports cannot be put in practice
- Has safety issues –suicidal ideas or plans, psychotic signs, severe psychomotor agitation or retardation
- Has not responded to fair treatment”

STATE VARIABLES: SUSTAINED_DEPRESSIVE_DISORDER
 ONGOING_PHARMA_STRATEGY
 DECISION VARIABLES: SUCCESSFUL_PHARMA_STRATEGY?
 ACTION VARIABLES: MEDICATION



4.4.2 CSI’s Hypertension Diagnosis and Treatment

The Clinical Algorithm provided by the Institute for Clinical Systems Improvement (ICSI) in Figure 4 that represents the processes of diagnosis and treatment of hypertension is translated to the SDA* notation. The result is:



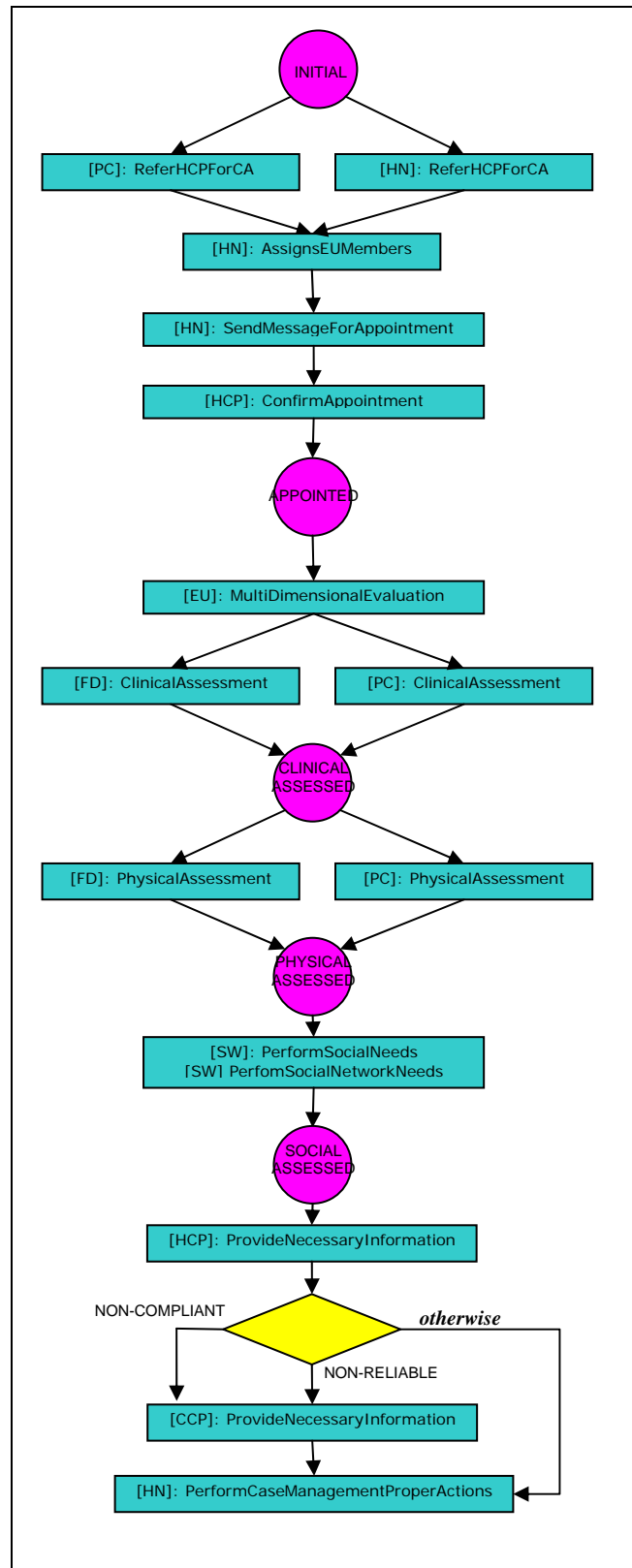
4.4.3 Comprehensive Assessment K4CARE Procedure

In the K4CARE healthcare model [2] *comprehensive assessment* is a service that comprises multi-dimensional evaluation plus clinical assessment and physical examination (integrating the medical side) and social needs and social network assessment (integrating the social side). It is the service devoted to detect the whole series of patient diseases, conditions and difficulties, from both the medical and social perspectives. This service is implemented with a procedure that may be represented in the SDA* model as it follows.

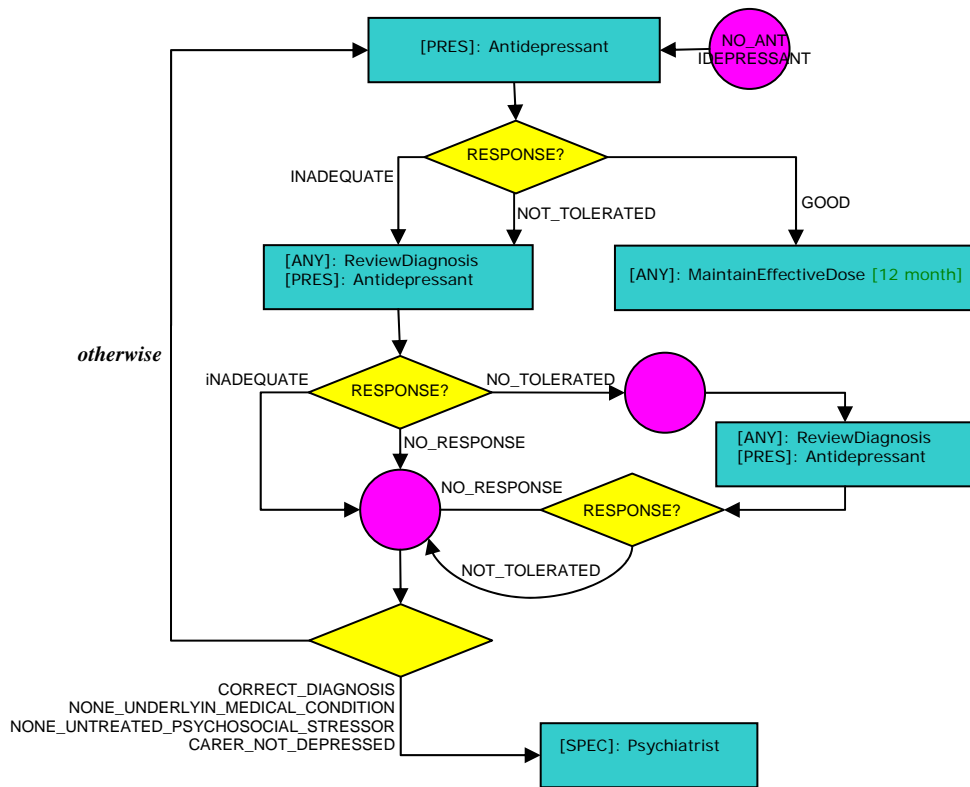
Here, the actions indicate the actor performing the action (i.e. performers) because comprehensive assessment is a collaborative process achieved with the combined actions performed by different actors.

[HCP]:	Action performed by the Home Care Patient.
[FD]:	Action performed by the Family Doctor.
[PC]:	Action performed by the Physician in Charge of the patient.
[HN]:	Action performed by the Head Nurse.
[SW]:	Action performed by the Social Worker.
[EU]:	Action performed by the Evaluation Unit (nuclear work team).
[CCP]:	Action performed by the Continuous Care Provider.
[ANY]:	Action performed by any of the K4CARE actors.

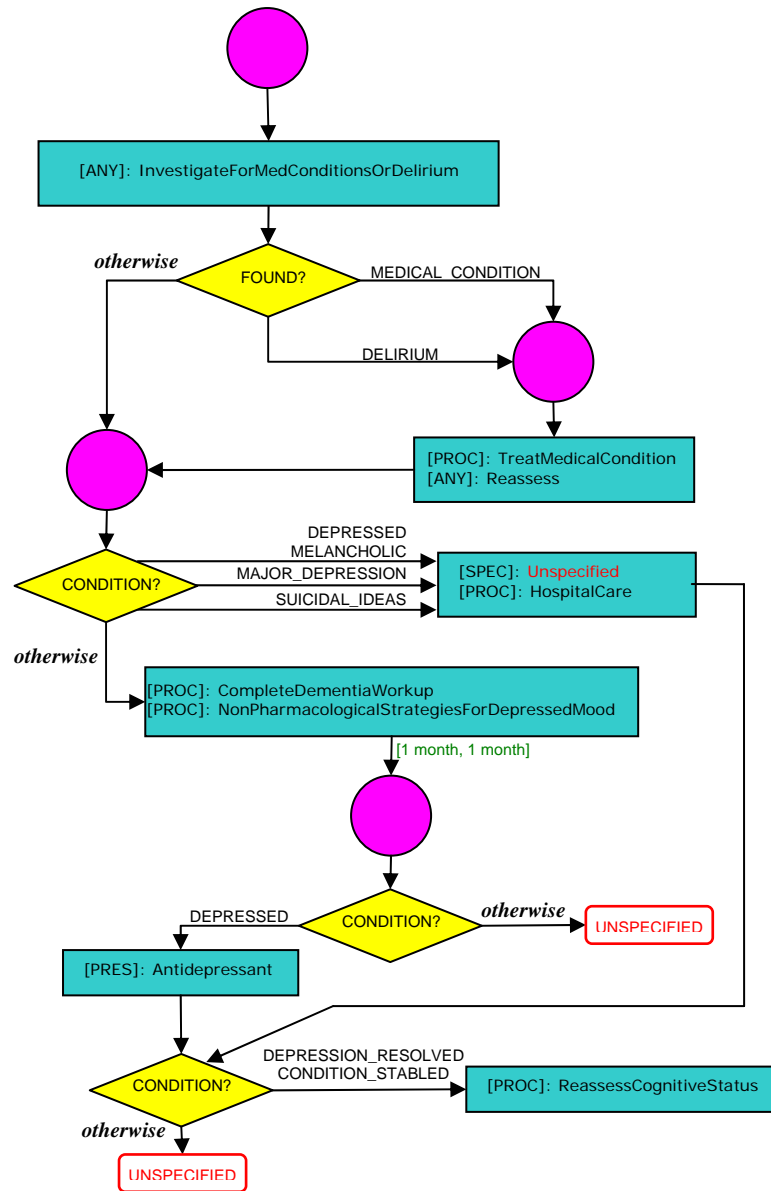
Table 8. Sorts of action variables for comprehensive assessment.



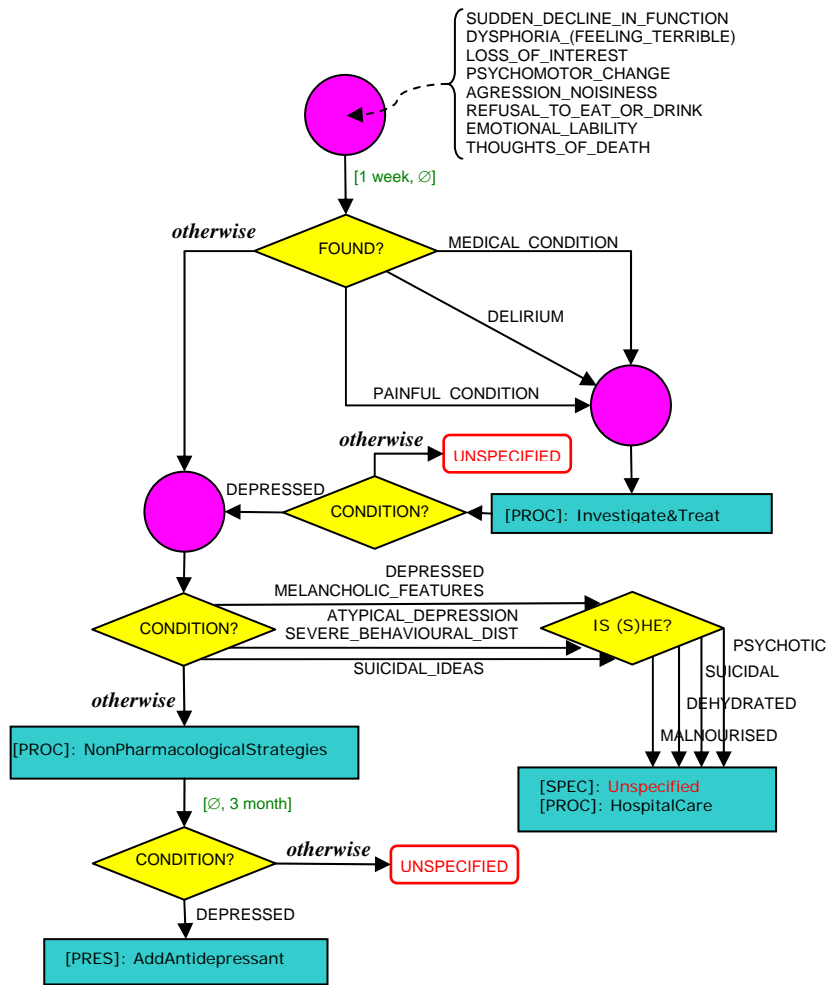
4.4.4 The use of Antidepressant Medication in the Elderly



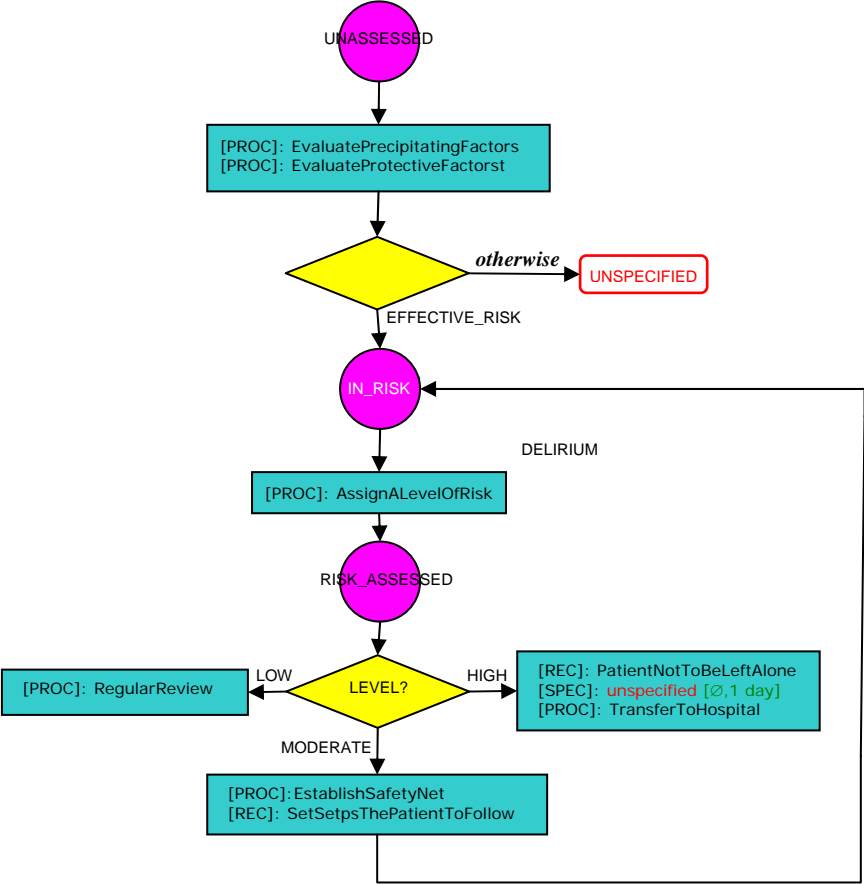
4.4.5 Management of Depression with Cognitive Impairment



4.4.6 Management of Depression with Dementia



4.4.7 Suicide: Risk of Assessment and Management



5 Conclusions and Acknowledgements

The SDA* model is a formal way of representing procedural knowledge in medicine. The first version of this model has been introduced in the paper and used to describe procedures and formal intervention plans in the K4CARE project. Parallel to the definition of the model, the application SDA Lab for the management of SDA* knowledge structures has been developed, though this tool is not discussed in the paper. The author wants to thank the work done by Joan Albert López in the development of the SDA Lab, to Miquel Millán and Montse Batet for providing support in the revision of the XML Schema, also to Dr. Fabio Campana for suggesting the guidelines and clinical algorithms used as examples in the document.

This work is part of the European Project K4CARE (IST-026968: Knowledge-Based Homecare eServices for an Ageing Europe) and the Spanish National Research Project HYGIA (TIN2006-15453-c04).

6 References

- [1] Bury J., Fox J., & Sutton D. The PROforma guideline specification language: progress and prospects. Proceedings of the First European Workshop, Computer-based Support for Clinical Guidelines and Protocols (EWGLP 2000), Leipzig 13-14 Nov. 2000.
- [2] Campana F., Riaño D., et al. D01: The K4CARE Model. 2007.
- [3] Consensus Guidelines for Assessment and Management of Depression in the Elderly. http://mhcs.health.nsw.gov.au/policy/cmh/publications/depression/depression_elderly.pdf
- [4] Fox J., Johns N. & Rahmanzadeh A. Disseminating Medical Knowledge-The PROforma Approach. *Artificial Intelligence in Medicine*, 14, 1998, 157-181.
- [5] Guttag J. Abstract Data Types and the Development of Data Structures. *Comm of the ACM* 20(6): 396-404, 1977.
- [6] K4CARE Project: <http://www.k4care.net>
- [7] Kosara R, Miksch S. Metaphors of Movement: A Visualization and User Interface for Time-Oriented, Skeletal Plans. *Artif Intell Med* pp. 111-131, 22(2), 2001.
- [8] Mor Peleg, PhD, Samson Tu, MS, Jonathan Bury, MBChB, Paolo Ciccarese, MSc, John Fox, PhD, Robert A. Greenes, MD, PhD, Richard Hall, MSc, Peter D. Johnson, MBBS, Neill Jones, MBBS, Anand Kumar, MBBS, Silvia Miksch, PhD, Silvana Quaglini, PhD, Andreas Seyfang, MSc, Edward H. Shortliffe, MD, PhD, and Mario Stefanelli, PhD. Comparing Computer-interpretable Guideline Models: A Case-study Approach. *J Am Med Inform Assoc*. 2003 Jan-Feb; 10(1): 52-68.
- [9] Shahar Y, Miksch S, Johnson P. The Asgaard Project: a task-specific Framework for the application and critiquing of time-oriented clinical guidelines. *Artif Intell Med* 1998;14:29-51.
- [10] http://www.asgaard.tuwien.ac.at/asbru_7_3/asbru_7.3_reference.pdf

APPENDIX: Schema **sda.xsd**

Complex types

sda_actionblock

sda_actionterm

sda_branch

sda_connector

sda_decision

sda_procedure

sda_state

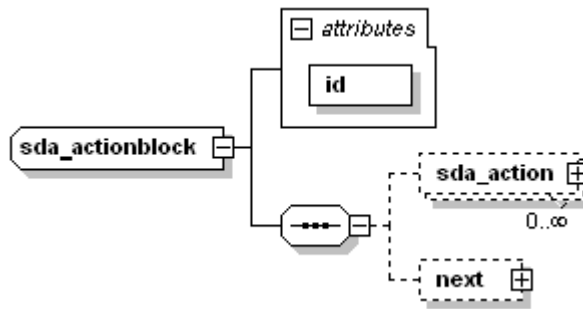
sda_term

Simple types

sda_time

complexType **sda_actionblock**

diagram



children **sda_action next**

used by element **sda_procedure/sda_action**

attributes	Name	Type	Use	Default	Fixed	annotation
	id	xs:string	required			

source

```
<xs:complexType name="sda_actionblock">
  <xs:sequence>
    <xs:element name="sda_action" type="sda_actionterm" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="next" type="sda_connector" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="required"/>
</xs:complexType>
```

attribute **sda_actionblock/@id**

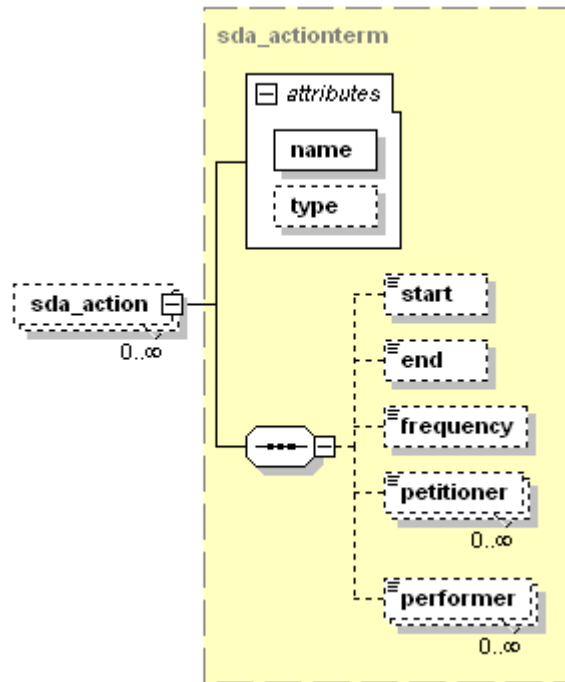
type	xs:string
properties	isRef 0
	use required

source

```
<xs:attribute name="id" type="xs:string" use="required"/>
```

element **sda_actionblock/sda_action**

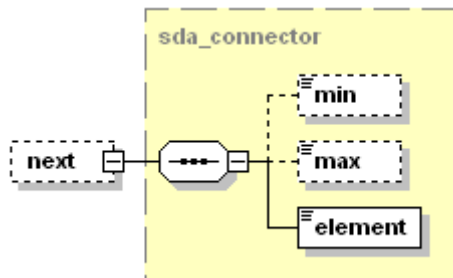
diagram



type	sda_actionterm					
properties	isRef	0				
	minOcc	0				
	maxOcc	unbounded				
	content	complex				
children	start end frequency petitioner performer					
attributes	Name	Type	Use	Default	Fixed	annotation
	name	xs:string	required			
	type	xs:string				
source	<xs:element name="sda_action" type="sda_actionterm" minOccurs="0" maxOccurs="unbounded"/>					

element **sda_actionblock/next**

diagram

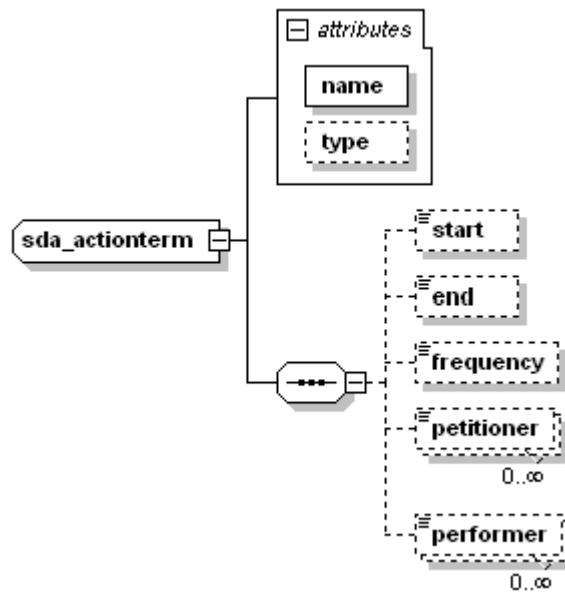


type **sda_connector**

properties	isRef	0
	minOcc	0
	maxOcc	1
	content	complex
children	min max element	
source	<code><xs:element name="next" type="sda_connector" minOccurs="0"/></code>	

complexType **sda_actionterm**

diagram



children	start end frequency petitioner performer					
used by	element	sda_actionblock/sda_action				
attributes	Name	Type	Use	Default	Fixed	annotation
	name	xs:string	required			
	type	xs:string				
source	<pre> <xs:complexType name="sda_actionterm"> <xs:sequence> <xs:element name="start" type="sda_time" minOccurs="0"/> <xs:element name="end" type="sda_time" minOccurs="0"/> <xs:element name="frequency" type="sda_time" minOccurs="0"/> <xs:element name="petitioner" type="xs:string" minOccurs="0" maxOccurs="unbounded"/> <xs:element name="performer" type="xs:string" minOccurs="0" maxOccurs="unbounded"/> </xs:sequence> <xs:attribute name="name" type="xs:string" use="required"/> <xs:attribute name="type" type="xs:string"/> </xs:complexType> </pre>					


attribute **sda_actionterm/@name**

type	xs:string	
properties	isRef	0
	use	required
source	<code><xs:attribute name="name" type="xs:string" use="required"/></code>	


attribute **sda_actionterm/@type**

type	xs:string
properties	isRef
source	<code><xs:attribute name="type" type="xs:string"/></code>

element **sda_actionterm/start**

diagram	
type	sda_time
properties	isRef 0 minOcc 0 maxOcc 1 content simple
facets	pattern [0-9]*[smhdwMy]
source	<code><xs:element name="start" type="sda_time" minOccurs="0"/></code>

element **sda_actionterm/end**

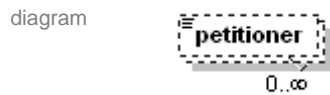
diagram	
type	sda_time
properties	isRef 0 minOcc 0 maxOcc 1 content simple
facets	pattern [0-9]*[smhdwMy]
source	<code><xs:element name="end" type="sda_time" minOccurs="0"/></code>

element **sda_actionterm/frequency**

diagram	
type	sda_time
properties	isRef 0 minOcc 0

	maxOcc	1
	content	simple
facets	pattern	[0-9]*[smhdwMy]
source	<code><xs:element name="frequency" type="sda_time" minOccurs="0"/></code>	

element **sda_actionterm/petitioner**



type	xs:string	
properties	isRef	0
	minOcc	0
	maxOcc	unbounded
	content	simple

source `<xs:element name="petitioner" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>`

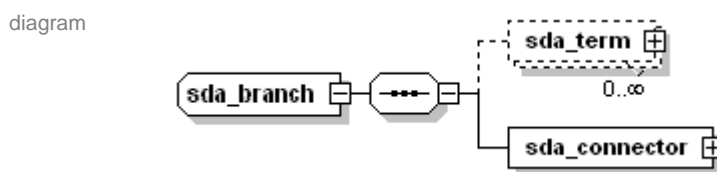
element **sda_actionterm/performer**



type	xs:string	
properties	isRef	0
	minOcc	0
	maxOcc	unbounded
	content	simple

source `<xs:element name="performer" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>`

complexType **sda_branch**

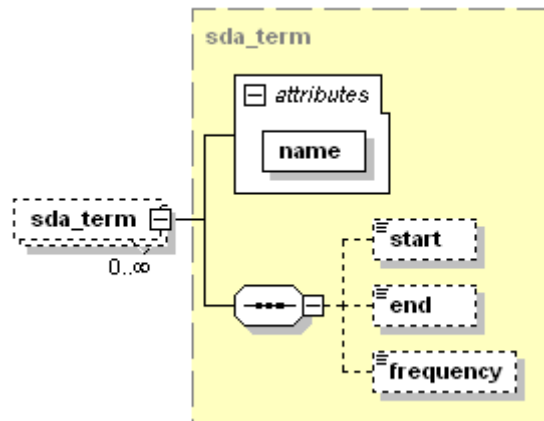


children	sda_term sda_connector	
used by	element	sda_decision/sda_branch

source `<xs:complexType name="sda_branch">
 <xs:sequence>
 <xs:element name="sda_term" type="sda_term" minOccurs="0" maxOccurs="unbounded"/>
 <xs:element name="sda_connector" type="sda_connector"/>
 </xs:sequence>
 </xs:complexType>`

element **sda_branch/sda_term**

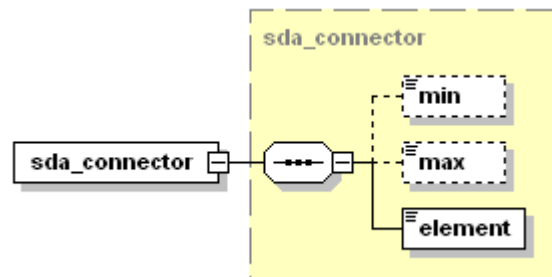
diagram



type	sda_term					
properties	isRef	0				
	minOcc	0				
	maxOcc	unbounded				
	content	complex				
children	start end frequency					
attributes	Name	Type	Use	Default	Fixed	annotation
	name	xs:string	required			
source	<code><xs:element name="sda_term" type="sda_term" minOccurs="0" maxOccurs="unbounded"/></code>					

element **sda_branch/sda_connector**

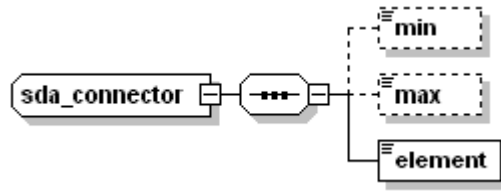
diagram



type	sda_connector					
properties	isRef	0				
	content	complex				
children	min max element					
source	<code><xs:element name="sda_connector" type="sda_connector"/></code>					

complexType **sda_connector**

diagram



children **min max element**

used by elements **sda_state/next** **sda_actionblock/next** **sda_decision/otherwise**
sda_branch/sda_connector

source

```
<xs:complexType name="sda_connector">
  <xs:sequence>
    <xs:element name="min" type="xs:time" minOccurs="0"/>
    <xs:element name="max" type="xs:time" minOccurs="0"/>
    <xs:element name="element" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

element **sda_connector/min**

diagram



type **xs:time**

properties

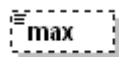
isRef	0
minOcc	0
maxOcc	1
content	simple

source

```
<xs:element name="min" type="xs:time" minOccurs="0"/>
```

element **sda_connector/max**

diagram



type **xs:time**

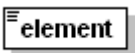
properties

isRef	0
minOcc	0
maxOcc	1
content	simple

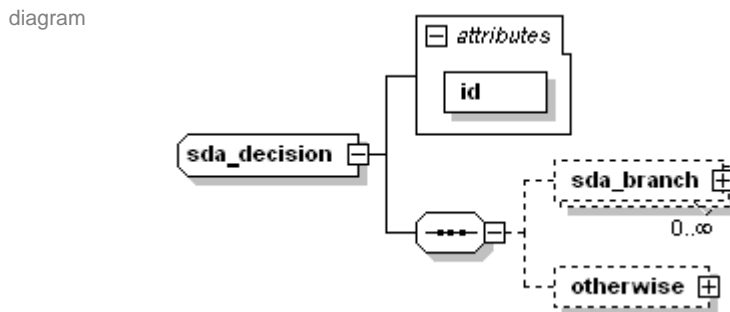
source

```
<xs:element name="max" type="xs:time" minOccurs="0"/>
```

element **sda_connector/element**

diagram		
type	xs:string	
properties	isRef	0
	content	simple
source	<code><xs:element name="element" type="xs:string"/></code>	

complexType **sda_decision**



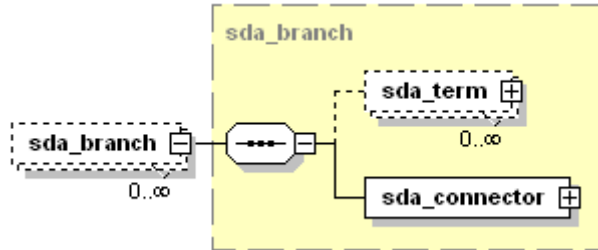
children	sda_branch otherwise					
used by	element	sda_procedure/sda_decision				
attributes	Name	Type	Use	Default	Fixed	annotation
	id	xs:string	required			
source	<pre> <xs:complexType name="sda_decision"> <xs:sequence> <xs:element name="sda_branch" type="sda_branch" minOccurs="0" maxOccurs="unbounded"/> <xs:element name="otherwise" type="sda_connector" minOccurs="0"/> </xs:sequence> <xs:attribute name="id" type="xs:string" use="required"/> </xs:complexType> </pre>					

attribute **sda_decision/@id**

type	xs:string	
properties	isRef	0
	use	required
source	<code><xs:attribute name="id" type="xs:string" use="required"/></code>	

element **sda_decision/sda_branch**

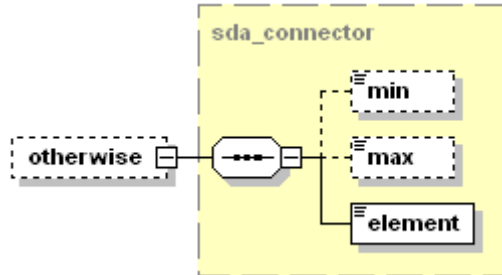
diagram



type	sda_branch	
properties	isRef	0
	minOcc	0
	maxOcc	unbounded
	content	complex
children	sda_term sda_connector	
source	<code><xs:element name="sda_branch" type="sda_branch" minOccurs="0" maxOccurs="unbounded"/></code>	

element **sda_decision/otherwise**

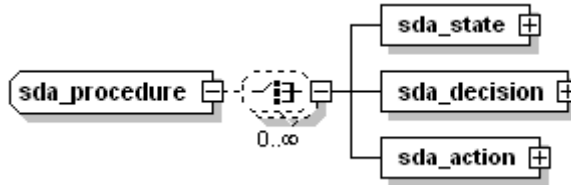
diagram



type	sda_connector	
properties	isRef	0
	minOcc	0
	maxOcc	1
	content	complex
children	min max element	
source	<code><xs:element name="otherwise" type="sda_connector" minOccurs="0"/></code>	

complexType sda_procedure

diagram



children

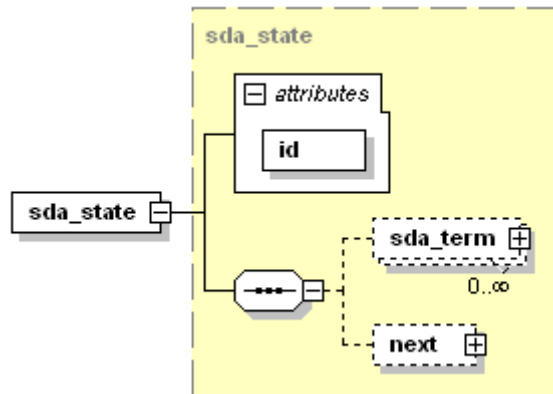
sda_state sda_decision sda_action

source

```
<xs:complexType name="sda_procedure">
  <xs:choice minOccurs="0" maxOccurs="unbounded">
    <xs:element name="sda_state" type="sda_state"/>
    <xs:element name="sda_decision" type="sda_decision"/>
    <xs:element name="sda_action" type="sda_actionblock"/>
  </xs:choice>
</xs:complexType>
```

element sda_procedure/sda_state

diagram



type

sda_state

properties

isRef

0

content

complex

children

sda_term next

attributes

Name

Type

Use

Default

Fixed

annotation

id

xs:string

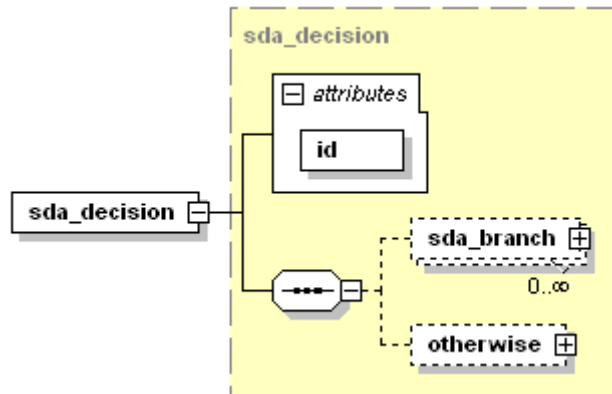
required

source

```
<xs:element name="sda_state" type="sda_state"/>
```

element `sda_procedure/sda_decision`

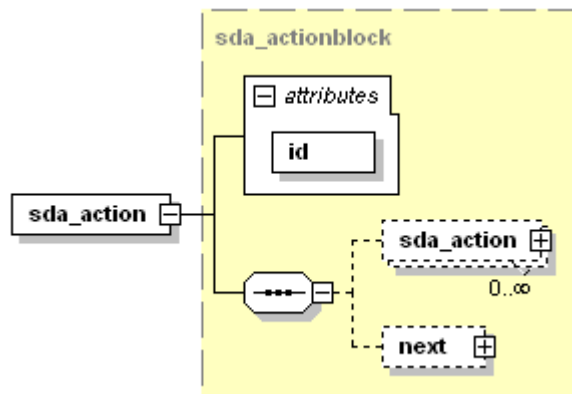
diagram



type	sda_decision					
properties	isRef	0				
	content	complex				
children	sda_branch otherwise					
attributes	Name	Type	Use	Default	Fixed	annotation
	id	xs:string	required			
source	<code><xs:element name="sda_decision" type="sda_decision"/></code>					

element `sda_procedure/sda_action`

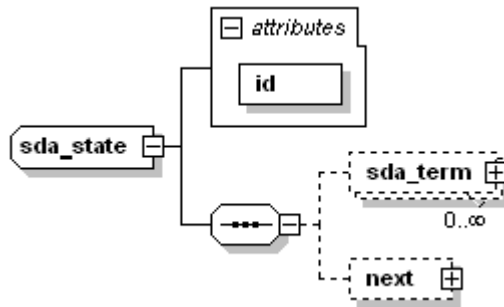
diagram



type	sda_actionblock					
properties	isRef	0				
	content	complex				
children	sda_action next					
attributes	Name	Type	Use	Default	Fixed	annotation
	id	xs:string	required			
source	<code><xs:element name="sda_action" type="sda_actionblock"/></code>					

complexType **sda_state**

diagram



children **sda_term next**

used by element **sda_procedure/sda_state**

attributes	Name	Type	Use	Default	Fixed	annotation
	id	xs:string	required			

source

```
<xs:complexType name="sda_state">
  <xs:sequence>
    <xs:element name="sda_term" type="sda_term" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="next" type="sda_connector" minOccurs="0"/>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="required"/>
</xs:complexType>
```

attribute **sda_state/@id**

type **xs:string**

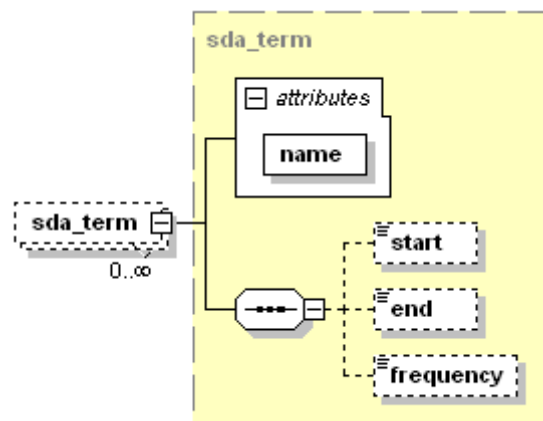
properties
isRef 0
use required

source

```
<xs:attribute name="id" type="xs:string" use="required"/>
```

element **sda_state/sda_term**

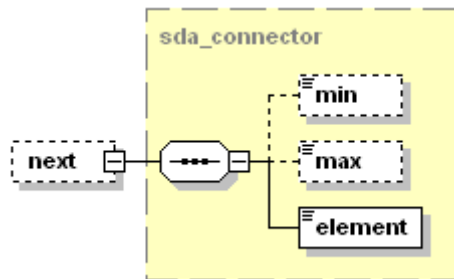
diagram



type	sda_term					
properties	isRef	0				
	minOcc	0				
	maxOcc	unbounded				
	content	complex				
children	start end frequency					
attributes	Name	Type	Use	Default	Fixed	annotation
	name	xs:string	required			
source	<code><xs:element name="sda_term" type="sda_term" minOccurs="0" maxOccurs="unbounded"/></code>					

element **sda_state/next**

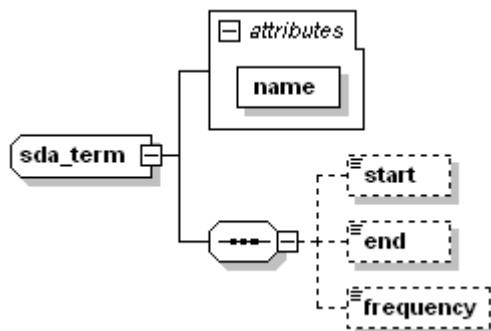
diagram



type	sda_connector					
properties	isRef	0				
	minOcc	0				
	maxOcc	1				
	content	complex				
children	min max element					
source	<code><xs:element name="next" type="sda_connector" minOccurs="0"/></code>					

complexType **sda_term**

diagram




children **start end frequency**

used by	elements	sda_branch/sda_term sda_state/sda_term				
attributes	Name	Type	Use	Default	Fixed	annotation
	name	xs:string	required			
source	<pre><xs:complexType name="sda_term"> <xs:sequence> <xs:element name="start" type="sda_time" minOccurs="0"/> <xs:element name="end" type="sda_time" minOccurs="0"/> <xs:element name="frequency" type="sda_time" minOccurs="0"/> </xs:sequence> <xs:attribute name="name" type="xs:string" use="required"/> </xs:complexType></pre>					


attribute **sda_term/@name**

type	xs:string	
properties	isRef	0
	use	required
source	<pre><xs:attribute name="name" type="xs:string" use="required"/></pre>	


element **sda_term/start**

diagram		
type	sda_time	
properties	isRef	0
	minOcc	0
	maxOcc	1
	content	simple
facets	pattern	[0-9]*[smhdwMy]
source	<pre><xs:element name="start" type="sda_time" minOccurs="0"/></pre>	

element **sda_term/end**

diagram		
type	sda_time	
properties	isRef	0
	minOcc	0
	maxOcc	1
	content	simple
facets	pattern	[0-9]*[smhdwMy]
source	<pre><xs:element name="end" type="sda_time" minOccurs="0"/></pre>	

element **sda_term/frequency**

diagram	
type	sda_time
properties	isRef 0 minOcc 0 maxOcc 1 content simple
facets	pattern [0-9]*[smhdwMy]
source	<code><xs:element name="frequency" type="sda_time" minOccurs="0"/></code>

simpleType **sda_time**

type	restriction of xs:string
used by	elements sda_term/end sda_actionterm/end sda_term/frequency sda_actionterm/frequency sda_term/start sda_actionterm/start
facets	pattern [0-9]*[smhdwMy]
source	<code><xs:simpleType name="sda_time"> <xs:restriction base="xs:string"> <xs:pattern value="[0-9]*[smhdwMy]"/> </xs:restriction> </xs:simpleType></code>