Towards an Automated Assistant for Clinical Investigations

Vivek Nigam
LMU, GERMANY
vivek.nigam@ifi.lmu.de

Tajana Ban Kirigin
University of Rijeka, HR
bank@math.uniri.hr

Andre Scedrov
UPENN, USA
scedrov@math.upenn.edu

Carolyn Talcott
SRI International, USA
clt@csl.sri.com

Max Kanovich
Queen Mary, University of London, UK
mik@dcs.qmul.ac.uk

Ranko Perovic
Senior Clinical Trial Specialist
perovicrankomd@gmail.com

Abstract

Before a drug can be made available to the general public, its effectiveness has to be experimentally evaluated. Experiments that involve human subjects are called Clinical Investigations (CIs). Since human subjects are involved, procedures for CIs are elaborated so that data required for validating the drug can be collected while ensuring the safety of subjects. Moreover, CIs are heavily regulated by public agencies, such as the Food and Drug Administration (FDA). Violations of regulations or deviations from procedures should be avoided as they may incur heavy penalties and more importantly may compromise the health of subjects. However, CIs are prone to human error, since CIs are carried out by the study team, which might be overloaded with other tasks, such as hospital and/or pharmacy duties, other trials, etc. In order to avoid discrepancies, we propose developing an automated assistant for helping all the parties to correctly carry out CIs as well as to detect and prevent discrepancies as early as possible. This way the proposed automated assistant would minimize error, and therefore increase the safety of the involved subjects. This paper takes the first steps towards that direction. In particular, we propose a model for collaborative systems with explicit time, called Timed Local State Transition Systems (TLSTS), and argue that it can be used for specifying procedures and regulations for CIs, which mention time explicitly. Finally we show how to implement a TLSTS specification using Maude, an existing computational tool based on rewriting.

Categories and Subject Descriptors H.4.0 [*Information Systems Applications*]:

Categories and Subject Descriptors Theory, Management

Keywords Formal Methods, Timed Collaborative Systems, Clinical Investigations

1. Introduction

There is little doubt that drugs have improved in several ways the quality of life of the population. One can now rely on a huge variety of drugs for different symptoms, from simple pain killers to drugs for diabetes or HIV patients. A great deal of this success is due to the careful understanding of the effectiveness of a drug through experimentation. Since drugs may affect people's health, before a drug is made available to the general public, one is required to perform experiments in order to determine its effectiveness. At the final stages of testing, one is often required to test the drug on human subjects. These procedures are called *Clinical Investigations* (CIs) [5]. Only when CIs are successfully carried out and the effectiveness of the drug is experimentally validated may the drug be approved by public agencies, *e.g.*, the Food and Drug Administration (FDA), to be made available to the general public.

There are two key concerns while carrying out CIs. The first concern is of assuring that the subjects' health is not compromised by the test. In order to protect the subjects' health, CIs are rigorously regulated and audited by (FDA) inspectors. Violations of FDA regulations may imply heavy penalties, both financial as well as of bad public relations.[1] The second key concern is to collect enough data to determine the effectiveness of the drug. Without such data, it is most likely that the drug will not be allowed by public agencies (such as the FDA) to be commercialized.

Normally, a pharmaceutical company (Sponsor) that wants its drug to be tested hires a Clinical Research Organization (CRO) which is specializing in carrying out CIs. Then a detailed plan, called a *protocol*, is elaborated by specialists explaining how CIs should be carried out in order to obtain the most conclusive results without compromising the health of the subjects involved. For instance, the protocol normally contains the number of subjects that need to be used in the CI, or the type of CI, *e.g.*, blind, double-blinded, comparison with placebo, etc, as well as the duration of the CI.[2] In order to carry out a CI, the Sponsor/CRO collaborate with health institutions, typically hospitals, which have the necessary means, *i.e.*, competent people and the equipment, to perform the clinical research and collect the required data. At each site, a Principal Investigator is assigned and is in charge of the clinical trial carried out at the site.

During a CI it is the responsibility of the Sponsor/CRO and the PI from the health institution to (1) make sure that there are no violations of the regulations imposed by public agencies, such as the FDA, which could lead to penalties, and (2) that there are no deviations from the protocol, which could lead to data that is not conclusive for determining the effectiveness of the drug. Deviations

IHI'12, January 28–30, 2012, Miami, Florida, USA.

[1] FDA maintains a list available online of the name of companies that have in the past violated regulations .

[2] To illustrate the complexity of some CIs, typical protocols are more than 100 pages long.

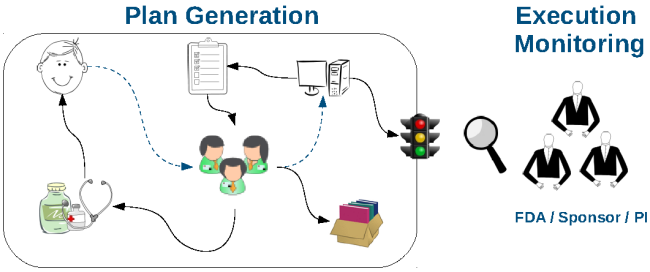

Figure 1. Diagram illustrating the two main applications of an automated assistant. Here the smiley figure is a subject, the green symbols the study team, the to-do list a plan, the lower-left symbols the medical procedures, the computer our assistant, while the box contains the physical files, *e.g.*, reports, and the three black suited agents are the FDA inspector, the Sponsor's monitor, and the health institution's PI.

could also compromise the safety of the subjects, which should be avoided at all costs. Furthermore, health institutions that have a record of deviations in previous CIs may be penalized by the market by not being hired for carrying out future CIs.

Although all participants of a CI, *e.g.*, physicians, study coordinators, pharmacists, and nurses, are informed of and even trained in the protocol and in the regulations, errors do occur. At each site it is common to have more than one ongoing trial, typically 5-10 trials, with different procedures, and they take place along with normal hospital duties. Furthermore, as all subjects are not enrolled in a study at the same time, it is difficult to have a clear overview of the process. Therefore, the process is prone to human error. It is easy to lose track, for example, that some subject has to be called for follow up or for an unscheduled visit. In order to avoid such errors, a monitor from the Sponsor/CRO that has in-depth knowledge of both the regulations and of the protocol monitors the on-going trials on site and *manually* checks the available records in the health institutions to identify situations that could lead to regulation violations or to protocol deviations. A much better approach would be to have an automated assistant that helps to correctly carry out CIs.

It has been shown that computer tools can be used to support carrying out administrative tasks. For instance, the tool Maude [2, 7], which we use in this work, has been successfully used for planning and workflow formalisms. Other formalisms include the PROforma clinical knowledge representation and workflow language and successors used for modeling health care processes, clinical guidelines, error handling and safe delegation [6]. However, to our knowledge there is no automated assistant on location that helps the medical team or the monitors in carrying out CIs.

For CIs, we identify two possible direct uses for such a tool. The first application is called *plan generation*. Plan generation preemptively avoids violations and deviations from occurring. For instance, once a nurse inputs a new test result in the system, she can be informed of the sequence of actions that need to be performed next, *e.g.*, carry out additional tests. The second application is called *execution monitoring*, that is, to check whether there have been violations and/or deviations in the past. When such violations or deviations are detected, the Sponsor/CRO/PI could react by, for instance, informing the corresponding authority, *e.g.*, FDA, and excluding incompetent sites or personnel from the CI or by excluding subjects from the trial.

Figure 1 illustrates how an automated tool could help to carry out CIs. For plan generation, when the subject arrives the health institution, the staff register the subject's visit in the automated tool. Then the tool generates a plan with the sequence of tasks that have to be performed during the visit. These tasks may include carrying out exams and providing drugs or filling out reports, such as safety reports. Moreover, whenever a task is finished, the staff may input the results in the tool and new plans might be generated. Since the staff is always informed by the tool of the actions that need to be taken, the chances of errors are reduced.

On the other hand, since the tool is aware of the actions taken by the staff, it can also detect when deviations and violations occur by execution monitoring. Once such events occur, the monitor or the PI can be informed, so that they can take the necessary actions, such as double-check the physical records. For instance, the tool could trigger an alarm sign (the traffic light sign) informing the responsible people that a problem has occurred. Such a tool could also be used by FDA inspectors in their first steps of an audit visit to have an overall view of the hospital procedures and the steps taken so far. The FDA audits can then decide which records and processes they want to take a closer look at.[3]

While we are still far from completely understanding the challenges of developing a full scale automated assistant, this paper provides the first steps in that direction. Our main contribution is to propose a model that can be used to specify regulations and protocols and that at the same time can be implemented in existing tools. Our contributions are summarized below:

- Section 2 illustrates, with existing FDA regulations, some of the types of properties that need to be expressed. We are particularly interested in properties that involve time, namely *explicit time intervals* and *past provisions*.
- The examples discussed in Section 2 serve as motivation for extending with explicit time our previous work on models for collaborative systems [8]. We introduce in Section 3 a mathematical model, called Timed Local State Transition Systems (TLSTS), which allows one to mention time explicitly, and argue by example that it can be used for specifying protocols and regulations.
- Moreover, in Section 4, we demonstrate that TLSTS specifications can also be easily implemented in Maude [2], an existing computational tool based on rewriting. Therefore, in principle, the construction of a prototype is within reach.

Finally, we conclude by pointing out the next steps of this project in Section 5. Further details can be found in the companion technical report [9].

2. Types of Properties

While studying the FDA regulations [5], we noticed that many clauses mention time explicitly. Such clauses served as motivation for extending with explicit time our previous work on models for collaborative systems [8]. We briefly describe two types of properties that we have identified, namely *explicit time intervals* and *past provisions*. Similar properties in the context of GLBA financial regulations are considered in [4]. (The emphasis in the quotes below is ours.)

Explicit Time Intervals Consider the following clause appearing in the Federal Regulations CFR21, Part 312 [5] for *Investigational New Drug Applications* (INDs).

> " (c) IND safety reports
>
> (1) Written reports –(i) The sponsor shall notify FDA and all participating investigators in a written IND safety report of:
>
> (A) Any adverse experience associated with the use of the drug that is both serious and unexpected; [· · ·] Each notification shall be made as soon as possible and *in no event later than 15 calendar days* after the sponsor's initial receipt of the information [· · ·]

[3] One could imagine more sophisticated provenance mechanisms that allow inspectors to even see on his monitor the electronic/scanned versions of the physical files.

(2) Telephone and facsimile transmission safety reports. The sponsor shall also notify FDA by telephone or by facsimile transmission of any unexpected fatal or life-threatening experience associated with the use of the drug as soon as possible but *in no event later than 7 calendar days* after the sponsor's initial receipt of the information."

The clause above mentions explicitly two different time intervals. The first is that one must send a detailed safety report to the FDA within 15 days, while the second obligation is that one must notify FDA of such an event within 7 days. These time intervals are used to specify *future obligations*, such as the obligation of sending a safety report.

Protocols mention explicit time intervals as well. For example, a protocol might specify that if some parameter of an urine test is three times above the upper limit, then the same test is repeated with a fresh sample *within 5 days* in order to make sure that the first result is not an isolated result.

Past Provisions Regulations and protocols also often enforce that to apply some action one must have satisfied some conditions in the past. A typical example of such requirement is that a subject can only participate in a CI if he has signed an informed consent. This is specified in Part 50 – Protection of Human Subjects, Subpart B, Section 50.20, quoted below:

" Except as provided in 50.23 and 50.24, no investigator may involve a human being as a subject in research covered by these regulations *unless the investigator has obtained the legally effective informed consent* of the subject or the subject's legally authorized representative. [$\cdots$] "

That is, a test can only be performed on a subject if he has signed an informed consent.

3. Timed Local State Transition Systems

This section introduces Timed Local State Transition Systems (TLSTSes) which is the mathematical model we propose for specifying regulations and protocols. As we demonstrate below, TLSTSes can specify the properties described above that mention time explicitly. TLSTS is a multiset-rewrite system that extends existing work [8] by adding explicit time and non-deterministic post-conditions to actions. Due to space limitations, we refer the formal definitions to the companion technical report [9] and just provide an intuition of how one can use such a model to specify CIs.

Timestamped Facts A *fact* is a ground, atomic predicate, *i.e.*, a predicate that does not contain variables. However, in order to accommodate time in our model, we associate to each fact a *natural number* called *timestamp*. *Timestamped facts* are of the form $p@t$, where t is the timestamp of the fact p. A timestamp of a fact can have several interpretations. For instance, the timestamped fact $consent(john, yes)@3$ denotes that the subject named John had signed the consent form at the third day. Alternatively, the fact $visit(2, id_{24}, no)@38$ denotes that the subject with identification id_{24} has an appointment scheduled on day 38.

Among the set of predicates, we distinguish the zero arity predicate $time$, which intuitively denotes the current global time of the system. For instance, the fact $time@2$ denotes that the global time is 2. A *state*, or *configuration* of the system is a finite multiset W of timestamped facts, and it contains exactly one occurrence of the predicate $time$. A configuration intuitively specifies the current state of the world. Consider the following configuration

$$\left\{\begin{array}{c} consent(john, yes)@3, sub(john, id_{24})@1, box(id_{24}, lbl)@1 \\ blind(Id, Lbl, placebo)@1, vital(1, id_{24}, yes)@10, \\ urine(1, id_{24}, high, no)@10, visit(1, id_{24}, yes)@10, \\ visit(2, id_{24}, no)@38, time@11 \end{array}\right\}$$

It specifies that John has already signed the consent form at day 3, that John's identification number is id_{24}, that boxes with labeled lbl are provided, and that these contain placebo and not the drug. Moreover, he has two scheduled visits, one at day 10 and another four weeks later at day 38, and John already appeared for the first visit and performed a vital signs test and the first urine test at the same day. The result of the urine test was high. Finally, it is the eleventh day of the CI.

Actions Actions behave like multiset rewrite rules and are used to change configurations by replacing facts. For example, the following action advances the time of a configuration:

$time@T \rightarrow time@(T+1)$.

When this action is applied to the configuration above, the fact $time@11$ is replaced by the fact $time@12$. Intuitively, it specifies the action of moving time forward. Our model allows for more complicated actions with time guards and multiple post-conditions. The following is an example of such action:

$$\begin{array}{r} time@T, urine(I, Id, bad, none)@T_1 \mid T_1 \leq T \leq T_1 + 5 \\ \rightarrow (time@T, urine(I, Id, bad, ok)@T) \\ \oplus (time@T, urine(I, Id, bad, high)@T) \\ \oplus (time@T, urine(I, Id, bad, bad)@T) \end{array}$$

Its pre-condition specifies that one should only repeat a urine analysis for a subject Id if the result of the first test was bad (three times the upper-limit) and no second test has been performed ($none$). Moreover, the guard of this rule specifies that one is only allowed to repeat this test in at most 5 days after the first test was performed, as specified by the protocol. Finally, it has three post-conditions, separated by the disjunction symbol $\oplus$. They specify that there are three possible outcomes of this action. Either the result of the second test is ok, $high$, or again bad. That is, the fact $urine(I, Id, bad, none)@T_1$ is replaced by one of the facts below:

$urine(I, Id, bad, ok)@T$, $urine(I, Id, bad, high)@T$,
or $urine(I, Id, bad, bad)@T$.

When specifying administrative systems, one normally makes use of identifiers that uniquely identify, for example, a person or a transaction. Identifiers are also used to anomymize a person's name. Before any test is performed on a subject, one assigns him a unique identifier, so that his results can be tracked down and not mixed up with the test of other subjects. As in our previous work [8], we specify such actions by using existential quantifiers ($\exists x$) in actions. For instance, the following action could be used to specify the action of assigning a unique identifier to a subject.

$$\begin{array}{r} time@T, sub(blank, N)@T_1, consent(N, yes)@T \\ \rightarrow \exists Id.[time@T, sub(Id, N)@T, consent(N, yes)@T, \\ scrTests(Id, blank)@T] \end{array}$$

Its pre-condition specifies that a subject, called N, has no identifier and that he already signed the consent form. Its post-condition specifies that a fresh value, Id, should be created and assigned to this subject. Moreover, the subject has to perform initial screening tests to check whether the subject is admissible for the CI.

The above action also illustrates how one can specify past provisions. For instance, as per FDA regulation Part 50, Subpart B, Section 50.20, a subject should only participate on the CIs if he signed a consent form. This is specified in the action above, as an identification number is only assigned to a subject if he has signed the consent form. For another example, a protocol might require that one first assigns an identifier to a subject before screening tests are performed, so that all his test results can be easily searched by using his identified. This condition is also expressed by the rule above. In particular, since the fact $scrTests(Id, blank)$ is first created by the rule above, the rule performing screening tests cannot be fired before a subject is assigned an identifier.

Timed Critical and Timed Goal Configurations Goal configurations specify states that one wants to reach, such as the successful execution of a CI, while critical configurations specify states

that we want to avoid, such as violations of the regulation and deviations from the protocol.

Formally, a timed goal configuration (respectively, timed critical configuration) is a pair formed by multiset of timestamped atomic predicates and a set of time constraints, $\langle \mathcal{M}, \mathcal{C} \rangle$. We say that a configuration $\mathcal{S}$ is a goal configuration (respectively, a critical configuration) if there is a ground substitution σ, such that $\mathcal{M}\sigma \subseteq \mathcal{S}$ and if all time constraints in $\mathcal{C}\sigma$ are true.

For a CI, a possible goal configuration is such that all visits of all subjects were correctly performed in the correct time, specified by the following goal configuration where we assume that there are 500 subjects.

$$\left\{ \begin{array}{c} visit(0, id_1, done)@T_{1,1}, \ldots, visit(24, id_1, done)@T_{24,1} \\ \cdots \\ visit(0, id_{500}, done)@T_{1,500}, \ldots, visit(24, id_{500}, done)@T_{24,500} \end{array} \right\}$$

and the set of time-constraints of the form

$T_{1,j} + (i-1) \times (28) - 5 \leq T_{i,j} \leq T_{1,j} + (i-1) \times (28) + 5$ for $1 \leq i \leq 24$ and $1 \leq j \leq 500$. The constraints specify that the time of the visits occurred every 4 weeks after the time of the first visit (also called baseline visit) $T_{1,j}$ with a tolerance of 5 days.

Timed critical configurations can be used to specify future obligations. For instance, the future obligation that a second urine test should be carried out in at least 5 days if the result of the first test was very high (three times the upper-limit) is specified by the following critical configuration:

$\langle \{time@T, urine(I, Id, bad, none)@T_1\}, \{T > T_1 + 5\} \rangle$.

If the second test has not been carried out within the next five days, then the configuration is critical, that is, it should be avoided.

For another example, the FDA regulation CFR21, Part 312 specifies that one must notify FDA if any serious or unexpected event has been detected within 7 days. This future obligation can be specified by critical configurations of the following form: $\{time@T, detected(Id, Num)@T_1, fda(ID, none, Num)@T_2\}$ with the time constraint $T > T_1 + 7$. This critical configuration specifies that a problem has been detected at time T_1, but the FDA was neither notified nor a safety report was sent in 7 days time.

Linear and Branching Plans and Decision Problems As already pointed out, a key concern for FDA inspectors as well as for PI/Sponsor/CRO is to try to avoid as much as possible any violations of regulation and deviations from the protocol. There are two *non-exclusive* ways of approaching this problem an *a priori* and/or an *a posteriori*. For the *a priori* approach, one can preemptively avoid violations and deviations by informing the participants of the actions they need to take. For the *a posteriori* approach, if a violation or a deviation is detected, the PI/Sponsor/CRO can take actions to counter-measure the situation. For example, the PI/Sponsor/CRO may need to inform FDA, so to avoid penalties, or exclude a subject from the CI, so to not compromise the test results. Correspondingly, we identify two different decision problems, *plan generation*, an *a priori* approach, and *execution monitoring*, an *a posteriori* approach.

First, we formalize the notion of *branching plans*. Since actions may result in different post-conditions due to the $\oplus$ connective, plans may branch. Intuitively, each branch corresponds to one post-condition resulting from applying an action. For instance, when a urine test is performed, there are three possible outcomes. According to the outcome, different plans must be followed. Formally plans are trees whose nodes are configurations. The nodes that are not leaf nodes are annotated by an action name act, while leaf nodes are not annotated by an action name. Moreover, the number of sub-trees of an annotated node, n, is exactly the number of post-conditions of act, each corresponding to a post-condition of applying the action act to n.

A branching plan considers all possible outcomes of an action and specifies actions to be taken for each possible case and sub-cases. In particular, we will be interested in plans that do not reach any critical configuration and that each branch reaches a goal configuration. These objects are necessary for the *plan generation* problem.

However, once an action is *actually* performed no branching occurs, but the resulting state is obtained by one of the post-conditions. For instance, when an urine test is carried out, then the result is either ok, $high$ or bad. Hence, the input for execution monitoring is not a branching plan, but a linear one, where the outcomes of all actions are already known. We call these plans *linear plans*. Formally, a linear plan is a *sequence* of configurations each annotated with an action name, act, and an natural number. The natural number specifies the post-condition of act used to obtain the next configuration in the sequence.

We are now able to formalize the problems of plan generation and execution monitoring as follows:

- **Generation of Branching Compliant Plans:** Given a TLSTS, an initial configuration, a set of timed goal configurations and a set of timed critical configurations, is there a branching plan that contains no critical configuration and that all its branches reach a goal configuration? Is it possible to generate such a plan?

- **Checking Compliance of an Execution:** Given a TLSTS, a set of timed critical configurations, and a linear plan, $\mathcal{P}$. Does $\mathcal{P}$ contain any critical configuration?

4. Implementation

In this section we briefly describe how the computational tool Maude [2] can be used to rapidly implement TLSTSes. An advantage of using Maude is that Maude is also based on rewriting. Therefore, the Maude encoding is very close to the actual TLSTS specification.

Configurations We start by specifying the signature of a TLSTS, *i.e.*, the set of constants and predicate symbols. For instance, the code below specifies that the zero arity operator `Time` is of sort (or type) `Fact` and that `Per` is a unary operator whose argument is of sort `Person`.

```
op Time : -> Fact .
op Per : Person -> Fact .
```

Other predicates of the sort `Fact` can be specified in a similar fashion.

Timestamped facts are specified by using the `@` operator which are used to attach a natural number to facts.

```
op _@_ : Fact Nat -> TFact .
```

To specify configurations as a multiset of timestamped facts, we first specify that timestamped facts is a subsort of configuration, denoted by the symbol `<`, that the empty set is a configuration, specified by the operator `none`, and that the juxtaposition of two configurations is also a configuration.

```
subsort TFact < Conf .
op none : -> Conf .
op _ _ : Conf Conf -> Conf [assoc comm id: none] .
```

The last line also specifies that configurations are multisets by attaching the keywords `assoc` and `comm`, which specify that the operator constructing configurations is both associative and commutative. Hence, when Maude checks whether an action (specified below) is applicable, Maude will consider all possible permutations of elements until it finds a match which satisfies the action's pre-condition as well as its guard. Finally, the keyword `id: none` specifies that the constructor `none`, specifying the emptyset, is the identity of an operator. For instance, it is used to identify the configurations `none (Time@2) none (Per(john)@3)` and `(Time@2) (Per(john)@3)`.

Timed Critical and Timed Goal Configurations Timed critical and timed goal configurations are specified as equational theories. For instance, the following equational theory specifies in

Maude the critical configuration when the FDA is not notified 7 days after a serious and unexpected problem is detected:

```
ceq critical(C:Conf time@T detected(Id,Num)@T1
    fda(Id,no,Num)@T2)= true if T > T1 + 7
```

Maude automatically replaces a configuration by `true` if it satisfies the condition specified by the equation above. Other critical configuration can be specifies in a similar way. Given this equational theory for critical configurations, one can determine whether a configuration `C` is critical by using an expression of the form `critical C == true`. This expression returns true if and only if the configuration `C` is critical. Similarly, an equational theory for timed goal configuration can be specified.

Actions with Multiple Postconditions Whereas critical and goal configurations are specified by using equational theories, actions are specified as rewrite rules in Maude. To accommodate branching plans, we need three new operators `noPlan`, denoting when a branching plan has no leaves, brackets used to mark a leaf of a plan, and `+` used to construct the list of leaves of a branching plan. The leaves of a branching plan belong to the sort `Plan`.

```
op noPlan : -> Plan .
op {_} : Conf -> Plan .
op _+_ : Plan  Plan -> Plan [assoc id: noPlan] .
```

The `+` is used to specify the different outcomes of an action. For instance, the following Maude rule specifies the action of performing a Urine test discussed in Section 3, where test has three possible outcomes, `ok`, `high`, or `bad`:

```
crl[urine]:
{(C:Conf time@T urine(1,24,bad,none)@T1)}
 => {(C:Conf time@T urine(1,24,bad,ok)@T)} +
  {(C:Conf time@T urine(1,24,bad,high)@T)} +
  {(C:Conf time@T urine(1,24,bad,bad)@T)}
 if T1 <= T ∧ T <= T1 + 5 .
```

The conditional rule above, labeled `urine`, specifies that when a urine test is applied then three different leaves are created, one for each possible result of an Urine test. The remaining facts appearing in the configuration `C` are left untouched. Moreover, the boolean operations at the end of the rule above specify the time constraints appearing in the corresponding action shown in Section 3.

With the machinery described above, one can specify an TLSTS using Maude. Once a system is specified, Maude provides powerful meta-level reasoning techniques which enable one to automatically check whether a linear plan is compliant or to construct compliant branching plans. Due to space limitations, we leave the details of this meta-level reasoning to the companion technical report [9]. There one can also find some samples of Maude code encoding a simplified CI scenario.

5. Future Work

To accomplish our ultimate goal of constructing a practical assistant tool for CIs, we identify the following tasks in both practical and theoretical domains. We plan to tackle them in the near future.

Decidability of the Plan Generation Problem – We identify foundational challenges to this project, such as complexity results for the plan generation problem. We already have some initial steps in this direction; namely, we have identified conditions on TLSTSes for which we believe that the plan generation is PSPACE-complete. The details can be found in the companion technical report [9]. Besides complexity results, we are also investigating implementation optimizations, such as ways to minimize search space.

Specify Complete Protocols – This paper proved the concept that it is possible to specify regulations and protocols. However, at the end, we would like to be able to specify complete protocols. In order to reduce the gap between the specification of protocols and its formalization, one might need to specify intermediate languages that are closer to the terminology and format used in protocols, but that is still precise enough to translate it to a TLSTS.

Human Computer Interface (HCI) – A crucial step on building a tool that is going to be at the end useful for the participants of a CI is to develop an intuitive HCI. We believe that our fruitful collaboration with CI specialists will continue to play a key role to accomplish this goal, in particular, in order to further understand the terminology and the procedures used.

Handling of Stored Data – Once a CI is carried out and data about subjects is collected, data is analyzed by specialist to determine the effectiveness of the drug. Since by using an automated assistant data is stored electronically, we expect to build interfaces to other existing tools which help specialists in their tasks by, for example, performing statistical analysis of data. This seems related to the on-going project described in [1] of connecting medical devices. On the other hand, as discussed in [3], this easier handling of data might also help lead to privacy violations. Since TLSTS is a model for collaborative system with privacy [8], we expect it to also be a suitable model for specifying and enforcing privacy policies.

Although we are years from achieving our ultimate goal, we believe that this is, nevertheless, a project worth investigating because of its theoretical and engineering challenges as well as of its potential impact on helping testing drugs, while taking care of the subjects' health. This project also shows that computer science and engineering have much to contribute on improving health practices.

Acknowledgments: This material is based upon work supported by the MURI program under AFOSR Grant No: FA9550-08-1-0352 and upon work supported by the MURI program under AFOSR Grant "Science of Cyber Security: Modeling, Composition, and Measurement". Additional support for Scedrov from NSF Grant CNS-0830949 and from ONR grant N00014-11-1-0555. Nigam was supported by the Alexander von Humboldt Foundation. Kanovich was partly supported by the EPSRC.

We thank Anupam Datta, Nikhil Dinesh, Insup Lee, John Mitchell, Grigori Mints, Oleg Sokolsky and Martin Wirsing for helpful discussion.

References

[1] D. Arney, M. Pajic, J. M. Goldman, I. Lee, R. Mangharam, and O. Sokolsky. Toward patient safety in closed-loop medical device systems. In ICCPS'10, 2010.

[2] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. *All About Maude: A High-Performance Logical Framework*. Springer, 2007.

[3] A. Datta, N. Dave, J. C. Mitchell, H. Nissenbaum, and D. Sharma. Privacy challenges in patient-centric health information systems. In *Usenix Workshop on Health Security and Privacy*, 2010.

[4] H. DeYoung, D. Garg, L. Jia, D. K. Kaynar, and A. Datta. Experiences in the logical specification of the HIPAA and GLBA privacy laws. In *WPES*, pages 73–82, 2010.

[5] FDA. Code of federal regulations, Title 21, Chapter 1, Subchapter D, Part 312: Investigational new drug application. Available at `http://www.accessdata.fda.gov/scripts/cdrh/cfdocs/cfcfr/CFRSearch.cfm?CFRPart=312`.

[6] M. A. Grando, M. Peleg, and D. Glasspool. A goal-oriented framework for specifying clinical guidelines and handling medical errors. *J. of Biomedical Informatics*, 2009.

[7] S. Iida, G. Denker, and C. Talcott. Document logic: Risk analysis of business processes through document authenticity. *J. of Research and Practice in Information Technology*, 2011.

[8] M. Kanovich, T. B. Kirigin, V. Nigam, and A. Scedrov. Bounded memory Dolev-Yao adversaries in collaborative systems. In *FAST*, 2010.

[9] V. Nigam, T. B. Kirigin, A. Scedrov, C. Talcott, M. Kanovich, and R. Perovic. Timed collaborative systems. `http://www2.tcs.ifi.lmu.de/~vnigam/docs/TR-TLSTS/`, June 2011.