Goal-oriented coordination with cumulative goals

Aditya Ghose[1], Hoa Dam[1], Shunichiro Tomura[1], Dean Philp[2], and Angela Consoli[2]

[1] Decision Systems Lab, University of Wollongong, Wollongong, NSW, 2500, Australia
{aditya, hoa}@uow.edu.au; st655@uowmail.edu.au
[2] Defence Science and Technology Group, Edinburgh, SA, 5111, Australia
{dean.philp, angela.consoli}@dst.defence.gov.au

Abstract. Multi-actor coordination is in general a complex problem, with inefficient and brittle solutions. This paper addresses the problem of abstract coordination of multiple actors at the goal-level, which often enables us to avoid these problems. Goals in multi-actor settings often rely on the cumulative efforts of a group of distinct actors whose individual, measurable contributions accumulate to lead to the fulfillment of the goal. We formalize these as *cumulative goals* in a mathematical framework that permits us to measure individual actor contributions on a variety of scales (including qualitative ones). We view these as *softgoals*, a conception general enough to incorporate the more traditional *hardgoals*. Within this mathematical setting, we address the coordination problem as one of deciding which actor is allocated to fulfilling each goal of interest (and the associated problem of goal model maintenance). We also report some empirical evaluation results.

1 Introduction

Coordination is often a complex problem, with inefficient and brittle solutions. Goals afford a convenient means for managing complex underlying systems by enabling the use of high-level abstractions specifying the intent underpinning the behaviour being described. While the importance of multi-actor goal modelling is well-recognized in the literature [1], little attention has been paid to the problem of multi-actor *coordination* at the goal level. This paper aims to address this gap. Our work views all goals as *softgoals*, which include *hard goals* as a special case. In our conception, goals may be relaxed (although we define specific conditions under which goal decompositions are valid) and provide directions for improvement (in a similar sense to non-functional requirements, such as the one for *usability*, which requires us to improve usability as much as possible within the applicable constraints).

Goals are organized in the form of (informally defined) AND-OR goal trees where goals higher up in the tree are decomposed (via AND-decomposition or

OR-decomposition) to lower level sub-goals. AND-decomposition involves situations where the parent goal is deemed to be fulfilled only if all of the sub-goals have been fulfilled. The correctness of an AND-decomposition is checked via three tests (originally articulated by Dardenne et al [2]): (1) the conjunction of the goal conditions of the sub-goals must entail the parent goal condition (the *entailment* requirement; (2) the conjunction of the sub-goal conditions together with the parent goal condition must be consistent (the *consistency* requirement); (3) no subset of the sub-goals should satisfy the above two requirements (the *minimality* requirement). OR-decomposition involves situations where the parent goal is fulfilled if any one of the sub-goals are fulfilled. The correctness of an OR-decomposition is ensured by checking that each that the goal condition of each sub-goal entails the goal condition of the parent goal.

We introduce the notion of a *cumulative goal*. Informally, a cumulative goal is one where the goal state specification includes a measurable parameter whose target value is achieved by the *cumulative fulfillment* of its sub-goals. Thus a goal to make a cup of tea, AND-decomposed into three sub-goals (boil water, pour it into a cup and dunk a teabag in the cup) is not a cumulative goal since there is no measurable parameter with a target value. On the other hand, a goal to deliver 30 tons of produce, AND-decomposed into 3 sub-goals (each involving the delivery of 10 tons of produce, by a different trucking service provider in each case) is an example of a cumulative goal (the measurable parameter with a target value is the amount of produce delivered). We use the abstract algebraic framework of *c-semirings* [3] to formalize cumulative goals. This provides a general framework where the parameters of interest can be measured not just in terms of numeric weights as in the example of above (the *weighted* instance of a c-semiring) but also using fuzzy, probabilistic and even qualitative scales. Bistarelli et al [3] also show that new c-semirings can be obtained from composing existing c-semirings.

We show that cumulative goals are *preferential* in the sense that they lead to *preference orderings* on states of affairs. This feature is particularly useful is viewing cumulative goals as softgoals (as an informal example, we prefer to achieve states s_1, failing which we prefer to achieve state s_2 and so on). We then outline the use of cumulative goals in multi-actor coordination. In the same way that goal-oriented modeling of behaviour is sequence-agnostic (it abstracts away detailed action sequencing), our approach also abstracts away action sequencing, task scheduling and so on. We define algorithms for goal model maintenance in settings where new actors become available (leading to new goals becoming feasible)

2 Cumulative goals

A c-semiring is a tuple $\mathcal{V} = \langle V, \oplus, \otimes, \bot, \top \rangle$ satisfying (for all $\alpha \in V$) [3]:

- V is a set of abstract values with $\bot, \top \in V$.
- $\oplus$ is a commutative, associative, idempotent and closed binary operator on V with $\bot$ as unit element ($\alpha \oplus \bot = \alpha$) and $\top$ as absorbing element ($\alpha \oplus \top = \top$).

- $\otimes$ is a commutative, associative and closed binary operator on V with $\top$ as unit element ($\alpha \otimes \top = \alpha$) and $\bot$ as absorbing element ($\alpha \otimes \bot = \bot$).
- $\otimes$ distributes over $\oplus$ (i.e., $\alpha \otimes (\beta \oplus \gamma) = (\alpha \otimes \beta) \oplus (\alpha \otimes \gamma)$).

We assume that each goal is written in the following format:

$\langle \phi, \{\langle \pi_0, v_0 \rangle, \ldots, \langle \pi_i, v_i \rangle, \ldots \langle \pi_n, v_n \rangle\}, C \rangle$

Here:

- ϕ is a formal assertion which the goal seeks to *achieve* or *maintain*
- Each of $\pi_0, \ldots, \pi_n$ is a parameter with a specified *target value*
- $v_0, \ldots, v_n$ are the target values for the parameters (v_0 for π_0 and so on)
- C is a collection of constraints, on a signature that is possibly a superset of the set of parameters referred to above, that specify permitted combinations of values of the parameters.

We shall refer to $\langle v_0, \ldots, v_n \rangle$ as the *level of satisfaction* of the goal.

We are now in a position to define an extended version of the rules governing the AND-reduction of goals [2].

A set of subgoals $\{G_0, \ldots, G_m\}$ is a valid AND-reduction of a goal G if and only if each of the following conditions are satisfied:

- $\phi_0 \wedge \ldots \wedge \phi_i \wedge \ldots \phi_n \models \phi$ (here ϕ is the formal assertion associated with goal G, ϕ_i the formal assertion associated with G_i and so on)
- $\phi_0 \wedge \ldots \wedge \phi_i \wedge \ldots \phi_n \wedge \phi \not\models \bot$
- There exists no subset of $\{\phi_0, \ldots, \phi_i, \ldots \phi_n\}$ which also satisfies the two conditions above
- For each i, $v_{i1} \otimes \ldots \otimes v_{in} \oplus v_i = v_{i1} \otimes \ldots \otimes v_{in}$, where v_i is the target value for parameter π_i in parent goal G, v_{ij} is the target value for parameter π_i is subgoal G_j and so on.

3 Cumulative goal achievement check

The comparison of actual observed values with their target values in goals reveals the goal satisfaction levels. Every goal receives the constraint check result in specific values. The aim of cumulative goal check is to investigate the constraint compliance level as the entire goal level by using the individual goal constraint check result.

The cumulative goal check can be accomplished in different frameworks by borrowing the idea of c-semiring [3]. One of the common approaches is the binary model, which can be formulated as below:

$$S_{CSP} =< \{0, 1\}, \vee, \wedge, 0, 1 > \tag{1}$$

where $\vee$ represents addition and $\wedge$ is multiplication. They are equivalent to "or" and "and" in the logic domain. The cumulative goals are satisfied only when all the individual goals satisfy their corresponding constraints. For example, if there are five goals g_1, g_2, g_3, g_4 and g_5, and the first four goals achieved a goal while g_5 could not, then 1 is assigned for the first four goals and 0 is allocated for the last one. In this case, the goals are not cumulatively achieved ($1 \wedge 1 \wedge 1 \wedge 1 \wedge 0 = 0$).

Due to its simplicity, this binary-based method can only express the result in a crisp manner. It cannot reflect the degree of cumulative goal achievements. Therefore, the fuzzy theory can be applied to overcome this issue, which is expressed as below:

$$S_{FCSP} =< \{x|x \in [0,1]\}, \max, \min, 0, 1 > \tag{2}$$

The one noticeable characteristic is the addition and multiplication operators. They are replaced with *max* and *min* operations. Another difference is the constraint achievement level expression in each goal, which can be expressed in any numbers between 0 and 1. The more a goal is close to the full achievement, the higher the result value becomes. 1 is provided if the cumulative goal is completely achieved. If g_1, g_2, g_3, g_4 and g_5 have 1, 0.6, 0.4, 0.8 and 0.9, then 0.4 is returned as the cumulative goal check result (*min* (1, 0.6, 0.4, 0.8, 0.9) = 0.4). This fuzzy theory based approach enables the system to determine the cumulative goal achievement level with non-crisp values.

This fuzzy approach can be extended by using weights, which can be defined as below:

$$S_{WCSP} =< \mathcal{R}^+, min, +, +\infty, 0 > \tag{3}$$

Respective goals entail cost, which is formulated as $(1 - fuzzy\ level) \times weight$ in a real number. The total cost is calculated by the summation of all cost from each goal. The optimum (set of) goals are the one which can suppress the cost at the minimum level. For example, we assume that we face with two goal achievement situations which have different variable values in each goal. The values for g_1, g_2, g_3, g_4 and g_5 in the first and second situations are [0.7, 0.6, 1, 0, 0.3], and [0.8, 0.9, 0.5, 1, 0] respectively. The cost for each goal is set as [3, 4, 1, 8, 6]. In this case, the total cost for the first situation is 14.7 ((1 - 0.7)×3 + (1 - 0.6)×4 + (1 - 1)×1 + (1 - 0)×8 + (1 - 0.3)×6 = 14.7), and that for the second situation is 7.5 ((1 - 0.8)×3 + (1 - 0.9)×4 + (1 - 0.5)×1 + (1 - 1)×8 + (1 - 0)×6 = 7.5). By comparing them with the *min* function, the situation two is considered as preferable due to less cost.

This c-semiring can even be expanded to quality domain. For example, temperature can be represented in categorical levels such as hot, warm, cool and cold. These temperature level achievement check in goals can be formulated as below:

$$S_{QCSP} =< \{x|x \in [cold, hot]\}, \max, \min, cold, hot > \tag{4}$$

For example, if we compare two goals g_1 and g_2 and each of them possess cold and cool respectively, then cool is returned if the addition operator is applied (*max*(cold, cool)). This is because cool is higher than cold in temperature. In contrast, cold is returned for the multiplication operator because the minimum value is returned (*min*(cold,cool)).The application of c-semiring to qualitative evaluation can facilitate the model to check the achievement level in a goal which cannot be expressed in a quantitative manner. In particular, non-functional softgoal achievement levels can be judged by this method because they can often be represented in qualitative conditions.

Another way to express the cumulative goal achievement level in a non-crisp manner is probability. One difference between the fuzzy-based approach and probability is the subjectivity. The level of fuzziness is determined by users through creating rules or thresholds while probability is measured based on collected facts and the value is objectively decided from the observed information. The probabilistic approach is formalized as follow:

$$S_{prob} =< \{x|x \in [0,1]\}, \max, \times, 0, 1 > \qquad (5)$$

The multiplication operator is substituted with the arithmetic multiplication . Each respective goal satisfaction level is determined from either of two different scenarios. If the goal is regraded to fully satisfy constraints from the observation, then 1 is allocated. In contrast, if the goal is not achieved, then the allocated value is 1 - $P(constraint)$. The goal constraint is treated as achieved only when the constraint itself is wrong. For example, if g_1, g_2, g_3, g_4 and g_5 are corresponded with their constraint achievement level 1, 1, 1, 0.2 and 0.2, then 0.04 is returned for the cumulative goal achievement level ($1 \times 1 \times 1 \times 0.2 \times 0.2 = 0.04$).

These different models can be integrated into one framework. There can be a case where one collection of goals needs to be expressed as the binary form while the other should be described in the fuzzy form. According to [3], the integration of different models can be presented as below:

$$S_{integration} = Comp(S_1, ..., S_n) \qquad (6)$$

S represents a model. By being able to combine different cumulative goal checks, the cumulative goal satisfaction level can be demonstrated in a more flexible way.

4 Cumulative preferential goals

The notion of a *softgoal* has long been recognized in the literature. Softgoals are goals that do not mandatorily have to be satisfied (as opposed to *hardgoals*) and admit some modicum of relaxation. They are typically used to model non-functional requirements, although they can also be useful in a range of other settings.

The conception of cumulative goal presented earlier in the paper allows us to view each such goal providing a preference ordering on a set of states.

A goal G of the form:

$$\langle \phi, \{\langle \pi_0, v_0 \rangle, \ldots, \langle \pi_n, v_n \rangle\}, C \rangle$$

leads to a preference ordering (most generally a partial pre-order) $\prec_G$ on the set of states. Let state s_1 be of the form:

$$\langle \phi_1, \{\langle \pi_j, v_j \rangle, \ldots, \langle \pi_s, v_s \rangle\}, C_1 \rangle$$

Let state s_2 be of the form:

$$\langle \phi_2, \{\langle \pi_k, v_k \rangle, \ldots, \langle \pi_t, v_t \rangle\}, C_2 \rangle$$

$s_1 \prec_G s_2$ if and only if:

- $\phi_1 \models \phi$ and $\phi_2 \models \phi$
- Each of C, C_1 and C_2 are individually satisfiable
- $\{\pi_0, \ldots, \pi_n\} \cap \{\pi_j, \ldots, \pi_s\} \cap \{\pi_k, \ldots, \pi_t\} = \{\pi_u, \ldots, \pi_r\}$
- $v_{u1} \oplus v_{u2} = v_{u2}.v_{r1} \oplus v_{r2} = v_{r2}$ and so on, where v_{u1} is the value associated with parameter π_u in state s_1, v_{u2} is the value associated with parameter π_u in state s_2 etc.

5 Coordination with cumulative goals

We work from the premise that dynamic nature of the operating context triggers the need for goal-oriented coordination. New agents with new capabilities may become available while existing agents might retire or otherwise become unavailable. The overall coordination model we work with assumes that there is a central coordinator agent that maintains the system-wide goal model while agents at the edge assume responsibility for the fulfillment of specific goals and opportunistically handover, or swap goals with other agents.

We outline below a procedure for maintaining goal models as new sub-goals become feasible to fulfill, or when existing goals or sub-goals become infeasible to fulfill. We use the notion of a *k-replacement goal model maintenance strategy*. In the case of new sub-goals becoming available, this involves taking any subset of size k of the set of new sub-goals and replacing a set of *sibling goals* (i.e., goals that share the same parent) with this set of size k. In the case of existing sub-goals becoming infeasible, this involves picking any subset of length k from the set of available sub-goals and seeking to replace the infeasible sub-goal(s) with this set. In the following, we use a function SLEVEL() which returns the *satisfaction level* (as defined earlier) for the goal provided as input.

We next outline a procedure to be used by agents to decide if a goal swap or handover is necessary. Recall that leaf-level goals are assigned to individual agents. These agents thus assume responsibility for fulfillment of these leaf-level goals.

6 Empirical evaluation

6.1 Goal achievement check performance

We embody the cumulative goal achievement check to show the achievability of our concept in reality. We implement six respective models introduced earlier; binary, fuzzy, weight, quality, probability and integration models. Through this

Algorithm 1 Agent coordination with central coordinator

```
Input: model: goal model, indicator: represented as ADDITION or REMOVAL,
    NEW: a set of new subgoals in case of ADDITION, REMOVE: the removal of
    a specific subgoal in case of REMOVAL, SUBGOAL-LIBRARY: for available sub-
    goals used in case of REMOVAL
Output: A new goal model
 1: if indicator=ADDITION then
 2:   for each non-root goal GOAL do
 3:     SAT-LEVEL := satisfaction level of GOAL
 4:   end for
 5:   for i= 1 To | $NEW$ | do
 6:     for each subgoal G of GOAL do
 7:       for each REPLACE = subset of size i of NEW do
 8:         set GOAL-TMP := GOAL with G replaced with REPLACE, provided
            the replacement satisfies the rules for valid AND-decompositions
 9:         if SLEVEL(GOAL-TMP) $\oplus$ SLEVEL(GOAL) = SLEVEL(GOAL-
            TMP) then
10:           set SAT-LEVEL := SLEVEL(GOAL-TMP) and GOAL := GOAL-
              TMP
11:         end if
12:       end for
13:     end for
14:   end for
15: else if indicator=REMOVAL then
16:   set GOAL to be the parent goal of the goal being removed
17:   set SAT-LEVEL := satisfaction level of GOAL
18:   for each REPLACE= subset of size i of NEW do
19:     set GOAL-TMP := GOAL with REPLACE added as one or more additional
        subgoals, provided the result satisfies the rules for valid AND-decompositions
20:   end for
21:   if SLEVEL(GOAL-TMP) $\oplus$ SLEVEL(GOAL) = SLEVEL(GOAL-TMP) then
22:     set SAT-LEVEL := SLEVEL(GOAL-TMP) and GOAL := GOAL-TMP
23:   end if
24: end if
```

experiment, we expect to assure the feasibility of the cumulative goal achievement check for all the respective models in a timely manner. The experimental result answers the question which expresses doubt in the realization of our ideas. This experiment consists of several steps.

1. **Model design**
   Each model is designed based on the definitions discussed in the previous sections. There are three main components in each model. The addition and multiplication are the operators which compute the achievement level of each individual and cumulative goals. Each operator is structured differently by following the definitions. Constraint check function computes the achievement level of each individual goal.

Algorithm 2 Goal swap/handover

Input: model: goal model, A: an agent with specified capabilities
Output: A possibly modified goal model. The possible modification would involve the assignment of the input agent to a leaf-level goal.

```
 1: set CANDIDATES := $\emptyset$
 2: for each leaf-level goal G currently assigned to agent A' do
 3:     if $\mathrm{SLEVEL}_A(G) \oplus \mathrm{SLEVEL}_{A'}(G) = \mathrm{SLEVEL}_A(G)$ then
 4:         set CANDIDATES := CANDIDATES $\cup$ {G}
 5:     end if
 6: end for
 7: for each distinct GOAL$\in$CANDIDATES do
 8:     for any DGOAL$\in$CANDIDATES currently assigned to agent A' do
 9:         if $\mathrm{SLEVEL}_A(\mathrm{GOAL}) \oplus \mathrm{SLEVEL}_{A'}(\mathrm{DGOAL}) = \mathrm{SLEVEL}_A(\mathrm{GOAL})$ then
10:             set CANDIDATES := CANDIDATES $-$ DGOAL
11:             non-deterministically select any goal from CANDIDATES and assign A to
                it
12:         end if
13:     end for
14: end for
```

2. **Goal generation**
10,000 goals are prepared for all the models except the integration model (the total number of goals is 40,000). Each goal possesses five target variables whose values are restricted with corresponding constraints. To simplify the experiment, target variable's constraint determines the desirable value as 1 for the binary, fuzzy, weight and probability models. The value is set as 3 for the quality model. In the integration model, 1 is the desirable value for the binary, fuzzy, weight and probability model parts while 3 is used for the quality model as well. Other constraints such as range and inequality are applicable. The achievement level of these individual goals determine the cumulative goals achievement level.
3. **Data generation**
Data is generated in order to simulate a real situation. For the binary, fuzzy and weight, we generate random values for each actual observed value from the range of 0 to 2. Each observed value in the generated data for the quality model is chosen from the integer number between 1 to 5. The probability model uses the data with variables assigned either 'A', 'B', 'C', 'D' or 'E'. The character is selected under uniform probability. The integration model uses these three dataset types since it combines all the different models. All the datasets consist of 10,000 records (goals) with eight attributes for the first dataset and five attributes for the second and third dataset. For the first dataset, the labels for each model are added as respective attributes beside the variables. The other datasets entail one label for the quality or probability model. The last row in all the datasets show the labels for the cumulative goal achievement level of each model. We create three datasets for each dataset type and hence nine different datasets are generated. In

total, we execute the same experiment with different datasets to increase the result reliability.

4. **Individual goal achievement check**
   Each goal's target variable is compared with the corresponding actual observed value. A returned result from the individual goal achievement check function differs from models.
   (a) **Binary**
   If a variable is equal to 1, then 1 is returned to the variable. Otherwise, 0 would be output. If all the observed variable values in a goal are the same with the target variable value (if all the variables receive 1), then 1 is returned to the goal. If one of the observed variables obtains 0 for the return, then 0 is returned to the goal as well.
   (b) **Fuzzy**
   A membership function is applied in the fuzzy model, which allows the model to represent the return value between 0 and 1 by providing a fuzzy level. The experiment sets all the fuzzy levels as 0.5. Any observed values larger than or equal to 0.5 and smaller than or equal to 1.5 return non zero values. If an observed variable value lies on the range, then the return can be expressed as a value larger than or equal to 0 and smaller than or equal to 1. The returned value depends on the achievement level of the value. If the observed variable is close to 1, it obtains a value close to 1, and vice versa. The fuzzy expression enables the model to determine goal achievement at a fuzzy level as well. Among the returned results from each target variable, the minimum value is selected as the representation of the goal's achievement level.
   (c) **Weight**
   Each variable possesses a weight. Five variables for each goal are set as 2, 4, 6, 8 and 10. Each goal obtains the total cost by summing all the possessing variables' weight. The multiplication of weight with corresponding variable's returned value reveals the cost. The returned value is decided based on the variables' values in a fuzzy way.
   (d) **Quality**
   Based on the variable numbers, the quality level of each variable is determined. Since the target value is 3 (moderate) and hence the value evaluation is based on the number 3. If the values are lower than 3, either low or cool is assigned depending on the value. In contrast, warm or hot is given to the number higher than 3. The values 1, 2, 3, 4, 5 correspond with cold, cool, moderate, warm and hot.
   (e) **Probability**
   Each variable receives either 1 or 1 - Prob(con) as return. We set 'A', 'B', 'C' and 'D' as allowing values which follow the constraints in this experiment. Thus, if a variable contains either of the characters, then the model considers the variable complies to the constraint and 1 is returned. However, if the variable possesses 'E', then the value 1 - Prob(con) is returned. In our experiment, we set the P(con) as 0.7. In reality, the probability depends on the information or criteria from a certain do-

main. The arithmetic multiplication of all the returned values from each variable leads to the achievement level of the goal.

(f) **Integration**
In this experiment, the integration consists of one binary, fuzzy, weight and probability models. For the binary, fuzzy and weight models, the same dataset is given. However, each model checks the individual goal achievement level based on its method. The probability model uses its own dataset and computes the achievement level. Hence, the total number of goals is 40,000. All the parameter settings are the same with the ones discussed in the (a) binary, (b) fuzzy, (c) weight and (d) probability subsections.

5. **Cumulative goal achievement computation**
Based on the result from the individual goal achievement level, the cumulative goal achievement level is calculated by the multiplication operation in each model. For the integration model, the cumulative result is shown as a collection of cumulative goals from each model.

The result from each model is measured by two metrics. Accuracy measures the matching rate of the actual computed results with the labels. Since the experiment is conducted three times with different datasets, we average the observed accuracy. If all the actual components from a model are identical with those from the expected result, the accuracy becomes 100%. For instance, the total labels for the binary model is 10,001 (10,000 goals plus 1 cumulative goal), and we check whether the returned values from the model are identical with the labels. If all the components' values are the same with those from the expectation, then the accuracy is 100%. When the model incorrectly computes the goal achievement level, then the returned result can differ from the expected result. This gap reduces the accuracy level. Large separation from the expected result leads to lower accuracy and indicates that it is infeasible to demonstrate our concept into actual implementation.

Another metric is time, which measures the computing time of each model. The calculation includes the individual and cumulative goal achievement check. It is vital to achieve the small time value because it allows the models to be used in real-time goal achievement checking. If the models spend a large amount of time, then it is unrealistic to be exploited in reality. We average the time value from each round. In the experiment, we use a 4GB memory, Intel(R) Core(TM) i5-7200U CPU, Surface Laptop. If the returned time is small although the device spec is not high, then we can demonstrate that the models are applicable to real applications.

The result of the experiment is shown in Table 1. As can be seen, the accuracy recorded as 100% for both individual and cumulative in all models, which validates the realization of our goal achievement level concept through the implementation. However, the accuracy is not supportive enough to conclude the achievability of our concept. The observation of recorded time is also required to check the ability of the models to check the goal achievement in real-time. From the Table 1, as the models become more complex (from the binary to integra-

Table 1. Accuracy and time from each model

	Accuracy		time (sec)
	Individual	Cumulative	
Binary	1	1	0.035
Fuzzy	1	1	0.079
Weight	1	1	0.098
Quality	1	1	0.052
Probability	1	1	0.102
Integration	1	1	0.363

tion), the total amount of time increases. However, all the observed time is below one second and remains in small values with 10,000 goals checking for the first four models and 40,000 for the integration model. This result demonstrates that our models can accurately check the goal achievement levels in a fast manner and can be used in real-time goal analysis if a single model is given.

6.2 Time increase analysis for integration model

In the previous experiment, the accuracy of each model is validated. In this experiment, we investigate the time increase rate for the integration model when the total number of integrated models increases. The aim of this experiment is to observe the possibility of the integration model being able to check the goal achievement level in a reasonable time manner when the total model numbers and goals rise. We analyze the effect of model number increase by different model types. We expect the time to increase with a linear pace in all the model types because other increase patterns such as polynomial and exponential growth can lead to prohibitive computational time.

We measure the time change by increasing each model up to five. Each added model checks the achievement level of additional 10,000 goals. Therefore, when five new models are added, the total goals number becomes 50,000. While increasing the total numbers of one model type, the others model types are removed. For example, in order to observe the time change by increasing the binary models, we augment the total number of the binary models in the integration model from one to five. When measuring the time difference, the other model types in the integration model are removed. We use three different datasets to average the observed time. All the details of parameters in each model type are the same with the ones in the previous experiment. The result is illustrated in Figure 1.

The binary model achieves the lowest time rise at 0.55 seconds when the number is five, followed by quality at around 0.3 and fuzzy at around 0.7 seconds. The weight and probability models recorded longer time compared to the others at approximately 0.6 for the former and 0.7 for the latter. One noticeable tendency is that though the slope of each line varies, their increase rates are in a linear form. We expect to observe a linear increase as discussed previously. The model computational time increase can be suppressed in a controllable level even though the number of any model types and goals increases. By combining both accuracy and time observed in both experiments, we can demonstrate that our

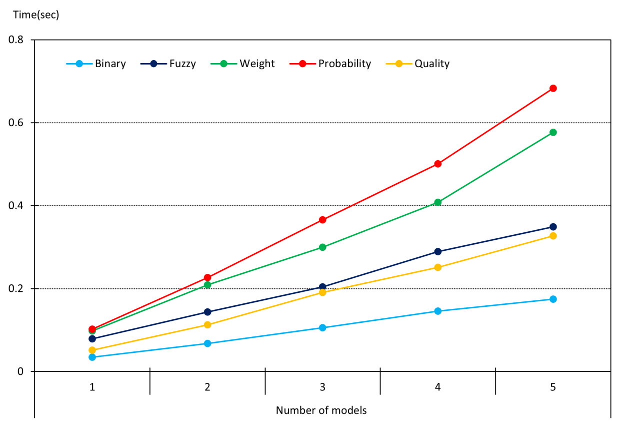

Fig. 1. Time change in integration model by increasing model numbers

concept is achievable to be utilized in reality under reasonable computational time.

7 Related Work

The versatility of the c-semiring allows the fusion of the concept with other fields. One example is argumentation formulation. [4] applies c-semiring based constraint check on Dung argumentation. The output of the constraint is treated as weights, which can be represented as either binary, fuzzy or other expressions depending on the requirements. These values are used as the attributes of edges between nodes as weights. The weights are served as attack levels to the directed nodes. This idea is materialized by the development of Conarg [5] [6]. This Java-based software enables checking conflict levels in argumentation formulation. The ability of the proposed method is deepened by defending from attacks by comparing two attacks from the opposite directions [7].

Another application is security. [8] applies the essence of c-semiring to maintaining desirable security degrees in multi-levels. One significant issue relating to security is the cascade elimination problems. The solution was considered an NP-hard problem. However, the generalization of this issue by using the c-semiring based approach reduces the difficulty into polynomial time level. The c-semiring approach also can be applied to the security protocol proposed by [9]. One substantial difference from the other approaches is that the c-semiring based approach allows to represent the protocol compliance level in a non-binary form by allowing constraint check result to express into non-crisp values such as fuzzy theory.

The goal oriented concept has been used in various realms as well. For instance, [10] extends the concept of i* as a text form, which is required to guarantee the scalability of the concept in a large project. Equipping i* with the ability of expression in a text form enables applying the concept into a complex problem. These instances assure the extensibility of the i* concept.

i* can even be utilized into the field of law constraint checking systems. [11] develops a model called Nomos, which allows the i* modeling to assure that any tasks entailing law constraints comply with the rules. For instance, tasks relating to clinical treatments are often restricted with a number of rules and requirements. These rules need to be represented together with the goals as well. Nomos is connected with VLPM [12] to be able to display the law constraints in UML diagrams.

8 Conclusions

References

1. Yu, E.S.: Towards modelling and reasoning support for early-phase requirements engineering. In: Requirements Engineering, 1997., Proceedings of the Third IEEE International Symposium on, IEEE (1997) 226–235
2. Dardenne, A.A., van Lamsweerde, A., Fickas, S.: Goal-directed requirements acquisition. Science of Computer Programming **20**(1-2) (1993) 3–50
3. Bistarelli, S., Montanari, U., Rossi, F.: Semiring-based constraint satisfaction and optimization. Journal of the ACM **44**(2) (1997) 201–236
4. Bistarelli, S., Santini, F.: A common computational framework for semiring-based argumentation systems. ECAI 2010: 19th European Conference on Artificial Intelligence **215** (2010)
5. Bistarelli, S., Santini, F.: Conarg: A constraint-based computational framework for argumentation systems. 2011 IEEE 23rd International Conference on Tools with Artificial Intelligence (2011) 605–612
6. Bistarelli, S., Santini, F.: Modeling and solving afs with a constraint-based tool: Conarg. International Workshop on Theorie and Applications of Formal Argumentation (2011) 99–116
7. Bistarelli, S., Rossi, F., Santini, F.: A novel weighted defence and its relaxation in abstractargumentation. International Journal of Approximate Reasoning **92** (2018) 66–86
8. Foley, S.N., Bistarelli, S., O'Sullivan, B., Herbert, J., Swart, G.: Multilevel security and quality of protection. Quality of Protection (2006) 93–105
9. Bella, G., Bistarelli, S.: Soft constraint programming to analysing security protocols. Theory and Practice of Logic Programming **4**(5-6) (2004) 545–572
10. Penha, F., Lucena, M., Lucena, L., Angra, C., Alencar, F.: A proposed textual model for i-star. iStar (2016)
11. Siena, A., Perini, A., Susi, A., Mylopoulos, J.: A meta-model for modelling law-compliant requirements. 2009 Second International Workshop on Requirements Engineering and Law (2009)
12. Villafiorita, A., Weldemariam, K., Susi, A., Siena, A.: Modeling and analysis of laws using bpr and goal-oriented framework. 2010 Fourth International Conference on Digital Society (2010)