DOI 10.1007/s00165-014-0318-7
BCS © 2014
Formal Aspects of Computing (2015) 27: 309–334

Formal Aspects of Computing

Abstraction and approximation in fuzzy temporal logics and models

Gholamreza Sotudeh[1] and Ali Movaghar[1,2]

[1] Department of Computer Engineering, Islamic Azad University (Science and Research Branch), Tehran, Iran
[2] Department of Computer Engineering, Sharif University of Technology, Tehran, Iran

Abstract. Recently, by defining suitable fuzzy temporal logics, temporal properties of dynamic systems are specified during model checking process, yet a few numbers of fuzzy temporal logics along with capable corresponding models are developed and used in system design phase, moreover in case of having a suitable model, it suffers from the lack of a capable model checking approach. Having to deal with uncertainty in model checking paradigm, this paper introduces a fuzzy Kripke model (FzKripke) and then provides a verification approach using a novel logic called Fuzzy Computation Tree Logic* (FzCTL*). Not only state space explosion is handled using well-known concepts like abstraction and bisimulation, but an approximation method is also devised as a novel technique to deal with this problem. Fuzzy program graph, a generalization of program graph and FzKripke, is also introduced in this paper in consideration of higher level abstraction in model construction. Eventually modeling, and verification of a multi-valued flip-flop is studied in order to demonstrate capabilities of the proposed models.

Keywords: Abstraction, Approximation, Fuzzy Kripke model, Fuzzy temporal logic, Model checking, Fuzzy program graph, Multi-valued flip-flop

1. Introduction

Model checking is an automatic formal technique for investigating correctness properties of systems, more specifically model checking also known as property checking refers to the following problem: "Given a finite-state model and a formal property, model checking automatically checks whether this property holds for that model" [BK]. Temporal properties of reactive systems can only be checked by temporal model checking. In these models, time is considered as either concrete (real-time) or discrete. In order to check the majority of real-time models such as timed automata (TA) [AD94], it is necessary to convert these models into discrete time models and then perform model checking process employing simpler logics such as computation tree logic (CTL). It is because logics defined alongside concrete time models such as timed computation tree logic (TCTL) are incapable of participating in model checking process.

Correspondence and offprint requests to: G. Sotudeh, E-mail: setoodeh@iaushiraz.ac.ir

Models that are meant to deal with different types of uncertainty can be verified through multi-valued temporal logics previously introduced in [FSGC05, CGD+06]. Fuzzy temporal logic is a variant of multi-valued temporal logic which is applicable in verifying some aspects of fuzzy systems. Expressions of different fuzzy temporal logics can be formed as formal, linguistic or even combinatorial expressions. Recent studies of using fuzzy logic in temporal model checking resulted in various trace checking approaches which are not suitable for system design phase. Moon et al. [MLL04] introduced a fuzzy temporal logic called fuzzy branching temporal logic (FBTL) with its corresponding model, which is highly applicable in model construction for fuzzy reactive systems, but they did not devise an automatic model checking approach. It seems such verification approach can only be achieved by converting FBTL's model into an intermediate discrete time model.

Considering the nature of fuzzy logic, it is necessary to quantize interval [0, 1] to a set of discrete values (each by a distance of Δ from another one) in favor of using model checking algorithms (and tools such as χChek). This requires the numbers participating in the model and propositions to be approximated using an approximation function such as rounding. In one hand, state space explosion occurs if Δ is small; on the other hand, the values of the propositions (that are belonging to [0, 1]) may change considerably if Δ is a large number. Therefore, it is necessary to prevent high deviation of proposition values by selecting Δ properly. So far, model/proposition approximation has not been studied while checking multi-valued models. The process of model/proposition approximation is one of the most important issues discussed in this paper. This process is similar to the concept of Fully Polynomial Approximation Scheme (FPTAS) [Vaz01] in designing approximate algorithms. For this reason, it is regarded as one of the initiatives in this paper.

In our series of research, using the concept of fuzzy logic, we propose to include different levels of uncertainty and inconsistency in model checking process of real-time systems. Finally we intend to introduce a model called fuzzy timed automata (FzTA) and its corresponding logic called fuzzy TCTL (FzTCTL) which is similar to the FBTL. The FzTCTL is not only intended to specify real-time fuzzy systems but also to perform model checking process on those systems. For this purpose, it is necessary to introduce discrete time models (and corresponding logics) along with an automatic verification approach (discrete models may be defined hierarchically). Final concrete model (i.e., FzTA) will be defined as a generalization of the proposed discrete models. Regarding the previous statement, it is possible for the resulted concrete model to be abstracted to the proposed discrete models.

In Sect. 3, an extended Kripke model, called fuzzy Kripke (FzKripke) is defined. The Kripke model is like the nondeterministic finite automata (NFA) in terms of expressing uncertainty, whereas the proposed FzKripke model is most similar to the fuzzy transition-nondeterministic finite automata (FT-NFA) [MSSY95]. In consideration of temporal properties verification, we defined the FzKripke's corresponding fuzzy temporal logic called Fuzzy Computation Tree Logic* (FzCTL*) in Sect. 4. In a nutshell, FzKripke and FzCTL* were formally defined by substituting the interval [0, 1] for the lattice of χKripke and χCTL while adding a series of accessory operators to χCTL. The concept of logic approximation on the proposed logic and its models is studied as a novel technique to encounter the state space explosion in Sect. 5. Model abstraction and abstracted model approximation is also studied in Sect. 6. A high-level abstraction (more flexible generalization) of the proposed FzKripke called fuzzy program graph (FzPG) is also introduced in Sect. 7. Model checking algorithms for discrete time models is already used as a verification tool for fuzzy systems (e.g. fuzzy controllers [IMMT05]); therefore, it is possible for the FzKripke and FzPG to be used in verification of fuzzy systems.

2. Background

2.1. Model checking

A powerful technique called model-checking had been proposed by Clarke in order to perform formal verification of temporal properties for reactive systems [CGP99]. In this technique, a mathematical and formal model (e.g. in the form of a graph) will be extracted from a system profile. A set of verification related propositions of the system called "properties" are also conformed to a formal language (denoted as temporal logic). Finally, in a mechanized approach, an algorithm is defined and implemented in order to exhaustively and automatically check properties of the model.

Table 1. Comparison of well-known temporal logics according to [BK, CGP99, HR04]

Language	Description
LTL	In LTL, "time" is found as a specific path (of states and events) and it is assumed that only one particular event occurs in each state of the path
CTL	Unlike the LTL, in CTL "time" has a branching behaviour, thus some events may occur after each state. For this reason, CTL paths are found as an infinite tree
CTL*	Like the CTL, this is a branching temporal logic, more specifically CTL* is a superset of CTL and LTL which freely combines path quantifiers and temporal operators

State space explosion This is one of the most challenging issues in model checking process. The problem is how to perform model checking process in proper time by handling a large number of states defined in a system model. For example, considering a graphical representation of a system model, the number of nodes in a model grows even to 10^{120} nodes in some cases. In order to encounter this problem, different techniques are used, one of which is symbolic checking of the model where compact data structure such as ordered binary decision diagram (OBDD) is applied to store the model. Another technique is abstracting larger models to the equivalent but more compact ones [BK, KP03, KNSW07, Wan06].

2.2. Modal logic

A type of formal logic that concentrates on the concept of modality in classical propositional and predicate logic is called modal logic. In this logic a statement may qualify regarding some conditions demonstrated by some modal (i.e., phrases that express modalities). Traditional modalities in classical logic are possibility, necessity and impossibility while modal logic is formalizing temporal modalities, modalities of knowledge, deontic and doxastic modalities.

Modal temporal logic Semantic of expressions with the qualification of "when" is called temporal logic. Some expressions are true all the times (i.e., tautologies) while others are sometimes true. In order to describe these temporal properties, formal languages (i.e., logics) such as CTL, linear-time temporal logic (LTL) and CTL* are innovated. These languages not only consist logical operators, but they also consist temporal adverbs such as "Next", "Finally", "Generally" and "Until". For this reason, these languages are called "Modal Temporal Languages".

Kripke model In the late 1950s and early 1960s Saul Kripke devised a novel model theory for non-classical logic systems; which is named Kripke semantics. In this model, there is a set of system states, each consists a set of atomic propositions. A transition edge (i.e., discrete event) exits from a particular state and enters to another state.

2.3. Temporal logic generalizations

Prior to this, various generalizations of temporal models and logics was presented such as, (1) generalizing discrete time to real time by defining a model named TA and its corresponding logic named TCTL, (2) considering probability to deal with uncertainty and random processes of systems, and (3) considering cost rather than time; there are many more combinations of these generalizations that are shown in [Gur03, RA11].

Multi-valued temporal logic These logics and their corresponding models are defined by substituting *boolean* algebra with *quasi-boolean* algebras. Generally in multi-valued temporal logics, a lattice is used instead of set $\{false, true\}$. These logics are applicable whenever different levels of uncertainty and inconsistency are required to be considered during the model checking process.

χKripke For each quasi-boolean algebra with a finite lattice such as $\mathbb{L}$, there is a model called χKripke such that values of atomic propositions and values of transition edges is a member of $\mathbb{L}$. χCTL is a multi-valued temporal logic corresponding to the χKripke model for discrete time, and χTCTL is its generalization over dense (concrete) time. There is also a tool named χChek to check χKripke models [ECD+03].

2.4. Fuzzy logic

Fuzzy logic is an infinite and continuous multi-valued logic that is used in various areas [Sla05, Wie10]. For instance, this logic is broadly used for designing intelligent and dynamic systems (e.g. controllers). These systems automatically respond to the produced events by gathering information. Since information gathering faces uncertainty and vagueness due to inaccuracy of monitoring infrastructure and uncertainty with information sources (e.g. network), response to events can no longer be done as crisp.

Temporal fuzzy logic Several studies were concentrated on temporal logics, related formal models, and related verification algorithms. Recently, few studies were made in relation to fuzzy logic and its applications in this community. Although fuzzy logic is a powerful mathematical tool for stating vagueness, it requires expansions to express temporal properties of uncertain dynamic systems [FPS12]. Such more perfect logics is called temporal fuzzy logic. These variety of logics can be used not only to check systems in early design phase, but also to monitor the behavior of systems during their execution.

Some variety temporal fuzzy logics used formal statements (i.e., fuzzy-time temporal logic [FPS12], FBTL [MLL04]) while the others used linguistic statements to declare their propositions (i.e., RELAX [WSB+09], fuzzy live adaptive goals for self-adaptive systems [BPS10]). Some other temporal fuzzy logics considered events to be discrete (i.e., fuzzy temporal formula [BG04], fuzzy propositional linear temporal logic [Pal00]) while others present events by a time interval in form of a fuzzy number, (i.e., FBTL [MLL04], RELAX [WSB+09], fuzzy live adaptive goals for self-adaptive systems [BPS10]). Assuming an execution trace of a particular system (a sequence of the performed events) in most of fuzzy temporal logics, it is only possible to test temporal properties of model by trace checking methods instead of using a formal model verification approach (i.e., fuzzy propositional linear temporal logic [Pal00], fuzzy temporal formula [BG04], fuzzy-time temporal logic [FPS12], fuzzy live adaptive goals for self-adaptive systems [BPS10]). However, trace checking methods facilitate the investigation of correctness check for some aspects of system execution, they are unable to perform integrated system checking. Therefore not all generalization of modal temporal logics can be used in the phase of system design phase. In the other words, only those logics whose models are derived from the whole system can be used in design phases (i.e., FBTL).

Fuzzy Branching Temporal Logic Moon et al. [MLL04] introduced one of the strongest fuzzy temporal logics called FBTL by generalizing CTL. Although this model is similar to the Kripke model; (1) its edges are defined as fuzzy values, and (2) time is considered to be concrete. In addition to adverbs of CTL, FBTL consists a set of comparison operations defined on time intervals. Among fuzzy temporal logics, FBTL is highly empowered in terms of expression and its model is very powerful, yet no model checking algorithm is developed for FBTL. This may be due to the computational complexity of the model and its inherent complexity. Given that verification of real time models (e.g. TA), first requires continuous to discrete time model conversion (e.g. Kripke) and then verification by simpler logics (e.g. CTL), the conjecture is defining simpler fuzzy logics and related models for model verification in FBTL is inevitable.

2.5. Fuzzy transition-nondeterministic finite automata

Mateescu et al. [MSSY95] introduced this finite fuzzy automaton for lexical analysis. Formally, a fuzzy automaton is a 5-tuple $(Q, q_0, \Sigma, \mathrm{F}, \delta)$, where Q is a finite set of machine states, q_0 is an initial state and a member of Q, Σ is the set of input symbols, δ is the transition function governing whether there is an existing transition between q_1 and q_2 as the elements of Q, and so is a mapping $\delta : Q \times I \times Q \to [0, 1]$, and finally F is the final state determination function defined as $\mathrm{F} : Q \to [0, 1]$. Unlike the Markov process [1], total value for outgoing edges of each node is not necessarily equal to 1. Actually each edge label in FT-NFA represents a value of possibility whereas in Markov process it represents a value of probability.

Example 1 An automaton is defined in the following example. In this automaton for each string belonging to the language, possibility of being a member in the string set is the highest obtained possibility for each accepting path. For instance, according to the automaton depicted in Fig. 1 occurrence possibility of "aba" through the "$q_1 q_1 q_1$" path is 0 but through "$q_1 q_2 q_3$" is equal to 0.6; thus the total possibility of occurring this string is 0.6.

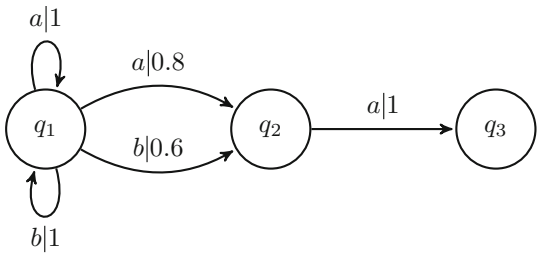


Fig. 1. An example of FT-NFA, as depicted above unlike the Markov process, total value for outgoing edges of each node is not necessarily equal to 1

3. Fuzzy Kripke model

This model is similar to χKripke model, except for its lattice; which is the continuous interval [0, 1]. This model is quite similar to the fuzzy graph defined in [Wie10], but in the terms of uncertainty it is most similar to the FT-NFA demonstrated in [RA11].

Model definition This model is defined as an ordered tuple, in the form of $M = (S, X, \mathrm{R}, \mathrm{L}, \mathrm{I})$ where $X = \langle x_1, \ldots, x_m \rangle$ is a set of attributes and $S = \{s_1, \ldots, s_n\}$ is a set of states. Every attribute has a possibility value; meanwhile different possible values for all attributes as a whole can be expressed by $\mathrm{Val}(X)$ as shown in (1).

$$\mathrm{Val}(X) = \{\langle v_1, \ldots, v_m \rangle | v_i \in [0, 1]\} \tag{1}$$

For attribute evaluation, access to the value of an attribute is defined by a dot operator as shown in (2).

$$\mu \in \mathrm{Val}(X), \quad \mu = \langle v_1, \ldots, v_m \rangle \Rightarrow v_i = \mu \cdot x_i \tag{2}$$

A function called L, assigns a label to each state as the state's specific valuation, see (3). Relation (4) defines R as a function that specifies the possibility of transition from one state to another. Entrance possibility for each state at start time can be represented by I a function that is defined in (5).

$$\mathrm{L} : S \to \mathrm{Val}(X) \tag{3}$$
$$\mathrm{R} : S \times S \to [0, 1] \tag{4}$$
$$\mathrm{I} : S \to [0, 1] \tag{5}$$

Following notation represents the transition between two states with a specific possibility.

$$s_i \xrightarrow{r} s_j \Leftrightarrow ((s_i, s_j), r) \in R$$

A finite execution path π starting from s'_0 can be defined as follows:

$$\pi \in Path_{fin}(s'_0) \Leftrightarrow \pi = s'_0 \xrightarrow{r_0} s'_1 \xrightarrow{r_1} s'_2 \xrightarrow{r_2} \cdots \xrightarrow{r_{u-1}} s'_u$$
$$\forall i \in 0 \ldots u_{i-1} \cdot r_i \in [0, 1]$$

The infinite execution path can be defined similarly:

$$\pi \in Path_{inf}(s'_0) \Leftrightarrow \pi = s'_0 \xrightarrow{r_0} s'_1 \xrightarrow{r_1} s'_2 \xrightarrow{r_2} \cdots$$
$$\forall i \in \mathbb{N} \cdot r_i \in [0, 1]$$

A certain state on a path and a sub-path can be defined by the following notation:

$$\forall i \in \mathbb{N} \cdot \pi[i] = s'_0 \wedge \pi[i..] = s'_i \xrightarrow{r_i} s'_{i+1} \xrightarrow{r_{i+1}} \cdots$$

4. Fuzzy CTL*

In order to define *Fuzzy CTL** (*FzCTL**) introducing some fuzzy operators is necessary. Fuzzy operators have a high variety of implementations, see [Sla05, Wie10], in this paper we use their simplest implementations whose properties are similar to the properties of quasi-boolean algebra related to χCTL*. The fuzzy operations are defined as follows:

1. $true = 1$
2. $false = 0$
3. $\neg a = 1 - a$
4. $a \sqcap b = \min(a, b)$
5. $a \sqcup b = \max(a, b)$
6. $a \rightarrow b = b \sqcup \neg a = \max(b, 1 - a)$

State propositions (as φ) and path propositions (as Φ) are expressible by the following grammar:

$\varphi ::= \mathrm{r} \mid \mathrm{x} \mid \neg\varphi \mid \varphi \sqcap \varphi \mid \varphi \geq \varphi \mid \wr\varphi + \varphi\wr \mid \mathrm{A}\,\varphi \mid \mathrm{E}\,\varphi$
$\Phi ::= \varphi \mid \neg\Phi \mid \Phi \sqcap \Phi \mid \Phi \geq \Phi \mid \wr\Phi + \Phi\wr \mid \mathrm{X}\,\Phi \mid \Phi\,\mathrm{U}\,\Phi$
$\mathrm{R} \in \mathbb{Q} \cap [0,1]$,
$\mathrm{x} \in \mathrm{X}$

Using the saturation operator [WwWG07], shown in (6), we have defined *bounded-add* $\wr\phi + \phi\wr$ operator. Although this operator is not seen in χCTL, it is defined in this logic.

$$\wr a\wr = \max(0, \min(1, a)) \tag{6}$$

Modal adverbs are also needed; the adverb "Finally" is defined as shown in (7), meanwhile "Generally" is defined in (8).

$$\mathrm{F}\Phi \overset{\text{def}}{=\!=} true\,\mathrm{U}\Phi \tag{7}$$

$$\mathrm{G}\Phi \overset{\text{def}}{=\!=} \neg\,\mathrm{F}\neg\Phi \tag{8}$$

A set of auxiliary operations may be defined in this grammar. The operators can be implemented using previously defined simple operators, each of which can be used for defining state propositions as well as path propositions.

- $\wr a - b\wr = \neg\wr\neg a + b\wr$
- $\mathrm{if}(a, b, c) = a \sqcap b \sqcup \neg a \sqcap c$
- $a \succcurlyeq b = \wr a + \neg b\wr$
- $a \preccurlyeq b = b \succcurlyeq a$
- $a \approx b = (a \succcurlyeq b \sqcap a \preccurlyeq b)$
- $a \not\approx b = \neg(a \approx b)$
- $a \succ b = \neg(b \succcurlyeq a)$
- $a \prec b = b \succ a$

Operators like $\{\succ \prec \succcurlyeq \preccurlyeq \approx \not\approx\}$ are called *quasi-comparison* and $\wr a - b\wr$ is called *bounded-subtract*. Definitions for other comparison operators are possible through the definition for operator $\geq$ in conjunction with logical operators.

Truth possibility of a proposition, when being in a state or on a path, can be presented by $\mathbb{P}$ and $\vDash$. In the following notations, the FzKripke model is represented by M while s represents a certain state in M, and π is an infinite path defined on M.

$$\mathbb{P}(M, s \vDash \varphi) \overset{\text{def}}{=\!=} \mathbb{P}(\varphi | M, s) \tag{9}$$

$$\mathbb{P}(M, \pi \vDash \Phi) \overset{\text{def}}{=\!=} \mathbb{P}(\Phi | M, \pi) \tag{10}$$

Semantic of state propositions can be defined as follows, where "op" may be a binary logical operator, a comparison operator, a quasi-comparison operator, or the bounded-add/subtract:

$$\mathbb{P}(M, s \vDash r) = r \tag{11}$$
$$\mathbb{P}(M, s \vDash x) = \mathrm{L}(s) \cdot x \tag{12}$$
$$\mathbb{P}(M, s \vDash \neg\varphi) = \neg\, \mathbb{P}(M, s \vDash \varphi) \tag{13}$$
$$\mathbb{P}(M, s \vDash \varphi \ \mathrm{op}\ \psi) = \mathbb{P}(M, s \vDash \varphi) \ \mathrm{op}\ \mathbb{P}(M, s \vDash \psi) \tag{14}$$

Konikowka and Penczek [KP03] articulated the inefficiency and ambiguity of $\vDash$ in expressing semantic of χCTL* formulas, therefore instead of using $\vDash$ we use $\vDash_A$ and $\vDash_E$ in following expressions.

$$\mathbb{P}(M, s \vDash \mathrm{A}\Phi) = \bigsqcap_{\pi \in Path_{inf}(s)} \mathbb{P}(M, \pi \vDash_A \Phi) \tag{15}$$
$$\mathbb{P}(M, s \vDash \mathrm{E}\Phi) = \bigsqcup_{\pi \in Path_{inf}(s)} \mathbb{P}(M, \pi \vDash_E \Phi) \tag{16}$$

Semantic of path propositions can also be defined follows, where "op" may be a binary logical operator, a comparison operator, a quasi-comparison operator, or the bounded-add/subtract:

$$\mathbb{P}(M, \pi \vDash_A \varphi) = \mathbb{P}(M, \pi \vDash_E \varphi) = \mathbb{P}(M, \pi[0] \vDash \varphi) \tag{17}$$
$$\mathbb{P}(M, \pi \vDash_\sigma \Phi \ \mathrm{op}\ \Psi) = \mathbb{P}(M, \pi \vDash_\sigma \Phi) \ \mathrm{op}\ \mathbb{P}(M, \pi \vDash_\sigma \Psi), \sigma \in \{\mathrm{A}, \mathrm{E}\} \tag{18}$$
$$\mathbb{P}(M, \pi \vDash_A \neg\Phi) = \neg\, \mathbb{P}(M, \pi \vDash_E \Phi) \tag{19}$$
$$\mathbb{P}(M, \pi \vDash_E \mathrm{X}\Phi) = \mathrm{R}(\pi[0], \pi[1]) \sqcap \mathbb{P}(M, \pi[1 \cdots] \vDash_E \Phi) \tag{20}$$
$$\mathbb{P}(M, \pi \vDash_A \mathrm{X}\Phi) = \mathrm{R}(\pi[0], \pi[1]) \rightarrow \mathbb{P}(M, \pi[1 \cdots] \vDash_A \Phi) \tag{21}$$
$$\mathbb{P}(M, \pi \vDash_\sigma \Phi\, \mathrm{U}\Psi) = \mathbb{P}(M, \pi \vDash_\sigma \Psi) \sqcup (\mathbb{P}(M, \vDash_\sigma \Phi) \sqcap \mathbb{P}(M, \pi \vDash_\sigma \mathrm{X}(\Phi\, \mathrm{U}\, \Psi))), \sigma \in \{\mathrm{A}, \mathrm{E}\} \tag{22}$$
$$\mathbb{P}(M, \pi \vDash_\sigma \mathrm{F}\Phi) = \mathbb{P}(M, \pi \vDash_\sigma \Phi) \sqcup \mathbb{P}(M, \pi \vDash_\sigma \mathrm{X}(\mathrm{F}\, \Phi)), \sigma \in \{\mathrm{A}, \mathrm{E}\} \tag{23}$$
$$\mathbb{P}(M, \pi \vDash_\sigma \mathrm{G}\Phi) = \mathbb{P}(M, \pi \vDash_\sigma \Phi) \sqcap \mathbb{P}(M, \pi \vDash_\sigma \mathrm{X}(\mathrm{G}\, \Phi)), \sigma \in \{\mathrm{A}, \mathrm{E}\} \tag{24}$$

Consequently the truth possibility of a proposition on the whole model should be defined as follows:

$$\mathbb{P}(M \vDash \varphi) \overset{\text{def}}{=\!=} \bigsqcap_{s \in S} (\mathrm{I}(s) \rightarrow \mathbb{P}(M, s \vDash \varphi))$$

Similar to χCTL which can be defined by restricting χCTL*, it is also possible to extract FzCTL from FzCTL*. Using the above-defined propositions along with the semantic of FzCTL correctness of the following equations can be investigated, where μ and ν stands for the greatest and smallest fixed-points, respectively [Tar55].

$$\neg\, \mathrm{AG}\, \varphi = \mathrm{EF}\, \neg\varphi \tag{25}$$
$$\neg\, \mathrm{AF}\, \varphi = \mathrm{EG}\, \neg\varphi \tag{26}$$
$$\neg\, \mathrm{A}(\varphi\, \mathrm{U}\, \psi) = \mathrm{EG}(\neg\psi) \sqcap \mathrm{E}(\neg\psi\, \mathrm{U}\, \neg\varphi \sqcap \neg\psi) \tag{27}$$
$$\mathrm{E}(\varphi\, \mathrm{U}\, \psi) = \mu\, Z \cdot (\psi \sqcup (\varphi \sqcap \mathrm{E}\, \mathrm{X}\, Z)) \tag{28}$$
$$\neg\, \mathrm{A}(\varphi\, \mathrm{U}\, \psi) = \nu Z \cdot (\neg\psi \sqcup (\neg\varphi \sqcap \mathrm{E}\, \mathrm{X}\, Z)) \tag{29}$$

Single-source FzKripke A fuzzy Kripke where I returns "1" for state s_0 and returns "0" for other states, is called *single-source FzKripke*. Every FzKripke $M = (S, X, \mathrm{R}, \mathrm{L}, \mathrm{I})$ can be converted to a single-source FzKripke $M' = (S', X, \mathrm{R}', \mathrm{L}', \mathrm{I}')$. For this purpose, a new state (such as ι) must be added as the source to S. For each state $s \in S$, equation $\mathrm{R}'(\iota, s) = \mathrm{I}(s)$ holds in the transition function of the new model and for the transition between each pair of states in S, functionality of R and R$'$ are identical. Function L$'$ will be considered such that:

1. For the source state, it returns "0" as the possibility of all attributes.
2. For other states it has the same functionality as L.

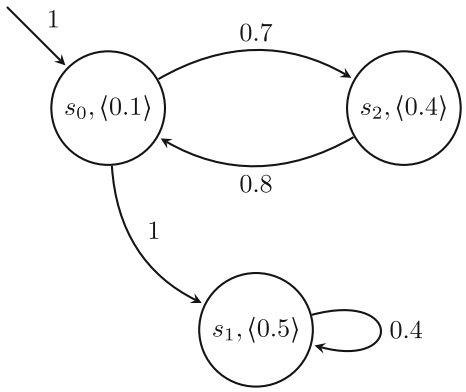


Fig. 2. FzKripke K_1 in which $X = x$ and function I returns 1 for s_0 and 0 for other states. Edges of K_1 are demonstrating the relation R while relation L is represented by decimals depicted in each state

Example 2 The following propositions hold for K_1 as shown in Fig. 2:

$$\mathbb{P}(K_1 \vDash \mathrm{EF}(x)) = 0.5$$
$$\mathbb{P}(K_1 \vDash \mathrm{EF}(\neg x)) = 0.9$$
$$\mathbb{P}(K_1 \vDash \mathrm{EG}(x < 0.5)) = 0.7$$

The functionality of I is defined according to the definition of single-source FzKripke model. In order to verify a property like φ on M, it is enough to check the proposition $\mathrm{AX}(\varphi)$ on a model like M'. M' with ι as its only source can also be represented as $M' = (S', X, \mathrm{R}', \mathrm{L}', \iota)$.

So far symbolic model checking methods are proposed for multi-valued Kripke models with a finite domain of values [CGD+06, Fai05, Gur03]. Having an infinite domain of values is the major obstacle to check fuzzy temporal logic on a fuzzy Kripke model. Even by having the ability of expressing fractional binary or decimal numbers using "T" digits in a computing system (e.g. a hardware system), we will face models with exponential state-space size depending on the factor "T". An important question is whether model checking with desired precision (with fewer significant digits) is possible in less amount of time and space or not. This problem will be discussed in the upcoming sections.

5. Approximability of FzCTL temporal properties on FzKripke

Participant numbers in fuzzy Kripke models and specification propositions are usually expressed in high precisions (i.e., several binary or decimal digits). Symbolic model checking of these models involves high costs in terms of time and memory, yet the results will be accurate. The cost can severely be degraded if the accuracy of the result decreases. In this section we demonstrate how to approximate the temporal properties of FzCTL on FzKripke. First we define a factor called *approximation error* denoted as ε such that 1 is divisible by ε, then by applying approximation function $\tau_\varepsilon(x)$ on model $M = (S, X, \mathrm{R}, \mathrm{L}, \mathrm{I})$, a new model like $M = (S, X, \mathrm{R}', \mathrm{L}', \mathrm{I}')$ is obtained in which, the following relations holds for each attribute $x \in X$ and each pair of s, s' in S:

$$\mathrm{R}'(s, s') = \tau_\varepsilon(\mathrm{R}(s, s'))$$
$$\mathrm{L}'(s) \cdot x = \tau_\varepsilon(\mathrm{L}(s) \cdot x)$$
$$\mathrm{I}'(s) = \tau_\varepsilon(\mathrm{I}(s))$$

If proposition φ is approximable then by approximating all of its constant numbers using function τ_ε we will obtain proposition φ_ε. Proposition φ is called *Simple ε-precision Approximable*, if it satisfies the following conditions:

1. It includes no comparison operator.
2. Right-hand side of any quasi-comparison operator, or bounded-add/subtract operator in φ, is a constant number that is dividable by ε.

Prior to investigate the approximability theorem, it is necessary to define some approximation functions. A "Discrete-Saturation" function is used to quantize real values with ε-precision and it is defined as shown in (30). A "Normal-Saturation" function for real values is defined as shown in (31). For a list of real values the function can be defined as shown in (32).

$$\wr x \int_{\varepsilon} = \wr \varepsilon \left\lfloor \frac{x}{\varepsilon} \right\rfloor \int = \max\left(0, \min\left(1, \varepsilon \left\lfloor \frac{x}{\varepsilon} \right\rfloor\right)\right) \tag{30}$$

$$[\![x]\!]_{\varepsilon} = \wr x \int_{\varepsilon} + \text{if}\,(\wr x \int = x, 0, 0.5\varepsilon) \tag{31}$$

$$\mu = \langle \nu_1, \ldots, \nu_m \rangle \Rightarrow [\![\mu]\!]_{\varepsilon} = \langle [\![\nu_1]\!]_{\varepsilon}, \ldots, [\![\nu_m]\!]_{\varepsilon} \rangle \tag{32}$$

Theorem 1 *If φ is a simple ε-precision approximable proposition, obtained by approximating the model and propositions of FzCTL* logic, the following statements are in place.*

A. *Using $\tau_{\varepsilon}(x) = \wr x + 0.5\varepsilon \int_{\varepsilon}$, then we will have $|\mathbb{P}(M, \varphi) - \mathbb{P}(M_{\varepsilon} \vDash \varphi_{\varepsilon})| < \varepsilon$.*

B. *Using $\tau_{\varepsilon}(x) = \wr x \int_{\varepsilon}$, then we will have $|\mathbb{P}(M, \varphi) - \mathbb{P}(M_{\varepsilon} \vDash \varphi_{\varepsilon})| < \varepsilon$.*

C. *Using $\tau_{\varepsilon}(x) = [\![x]\!]_{\varepsilon}$, then we will have $|\mathbb{P}(M, \varphi) - \mathbb{P}(M_{\varepsilon} \vDash \varphi_{\varepsilon})| < \frac{\varepsilon}{2}$.*

Proof. First we prove the theorem for FzCTL propositions. For further readability, we use $\varphi[s]$ instead of $\mathbb{P}(M, s \vDash \varphi)$ and $\varphi'[s]$ instead of $\mathbb{P}(M_{\varepsilon}, s \vDash \varphi_{\varepsilon})$ in this proof. Inductively we start from short propositions and extend it to long ones.

- For numbers we have:

  1. $\left| r - \wr r + 0.5\varepsilon \int_{\varepsilon} \right| < \varepsilon$ 2. $\left| r - \wr r \int_{\varepsilon} \right| < \varepsilon$ 3. $\left| r - [\![r]\!]_{\varepsilon} \right| < \frac{\varepsilon}{2}$

- For $\neg$ we have:

  1. For cases A and B we have:

  $$\left|\varphi[s] - \varphi'[s]\right| < \varepsilon \Rightarrow \left|(1 - \varphi[s]) - (1 - \varphi'[s])\right| < \varepsilon \Rightarrow \left|(\neg\varphi[s]) - (\neg\varphi'[s])\right| < \varepsilon$$

  2. For case C we have:

  $$\left|\varphi[s] - \varphi'[s]\right| < \frac{\varepsilon}{2} \Rightarrow \left|(1 - \varphi[s]) - (1 - \varphi'[s])\right| < \frac{\varepsilon}{2} \Rightarrow \left|(\neg\varphi[s]) - (\neg\varphi'[s])\right| < \frac{\varepsilon}{2}$$

- For $\sqcap$ we have:

  1. For cases A and B we have:

  $$\left|\varphi[s] - \varphi'[s]\right| < \varepsilon, \quad \left|(\psi[s]) - (\psi'[s])\right| < \varepsilon \quad \Rightarrow \left|(\varphi[s] \sqcap \psi[s]) - (\varphi'[s] \sqcap \psi'[s])\right| < \varepsilon$$

  Now if $\varphi'[s] < (\psi'[s] - \varepsilon)$ we have:

  $$\varphi[s] < \psi[s] \Rightarrow \left|(\varphi \sqcap \psi)[s] - (\varphi' \sqcap \psi')[s]\right| = \left|\varphi[s] - \varphi'[s]\right| < \varepsilon$$

  If $\varphi'[s] = \psi'[s]$, the value of $(\varphi \sqcap \psi)[s]$ will be equal to $\varphi[s]$ or $\psi[s]$ and correctness of the theorem can be investigated. If $\varphi'[s] = (\psi'[s] - \varepsilon)$, when $\varphi[s] \leq \psi[s]$ holds the proof is straight forward, but whenever $\varphi[s] > \psi[s]$, then $\psi'[s] > \varphi[s] > \psi[s] > \varphi'[s]$ and again the distance is less than ε.

  2. For case C we have:

  $$\left|\varphi[s] - \varphi'[s]\right| < \frac{\varepsilon}{2}, \quad \left|(\psi[s]) - (\psi'[s])\right| < \frac{\varepsilon}{2} \quad \Rightarrow \left|(\varphi[s] \sqcap \psi[s]) - (\varphi'[s] \sqcap \psi'[s])\right| < \frac{\varepsilon}{2}$$

  Now if $\varphi'[s] < (\psi'[s] - \frac{\varepsilon}{2})$ we have:

  $$\varphi[s] < \psi[s] \Rightarrow \left|(\varphi \sqcap \psi)[s] - (\varphi' \sqcap \psi')[s]\right| = \left|\varphi[s] - \varphi'[s]\right| < \frac{\varepsilon}{2}$$

If $\varphi'[s] = \psi'[s]$, the value of $(\varphi \sqcap \psi)[s]$ will be equal to $\varphi[s]$ or $\psi[s]$ and correctness of the theorem can be investigated. If $\varphi'[s] = (\psi'[s] - \frac{\varepsilon}{2})$, the proof is straightforward.

- For $\sqcup$, the proof is similar to $\sqcap$.
- Operator $\rightarrow$ is obtained by the combination of $\neg$ and $\sqcup$.
- For bounded-add in form of $\wr\varphi + r\wr$, we have:

  1. For cases A and B we have:

  $$\left|\varphi[s] - \varphi'[s]\right| < \varepsilon \Rightarrow \left|\wr\varphi + r\wr[s] - \wr\varphi' + r\wr[s]\right| < \varepsilon$$

  Notice r and 1 are divisible by ε and indeed $\neg r$ is divisible by that too. Here, we consider three cases:

  (a) If $\varphi'[s] < \neg r$ then $\varphi'[s] \leq (\neg r - \varepsilon)$ and since $\varphi[s] < \varphi'[s] - \varepsilon$, we have $\varphi[s] < \neg r$ and thus:

  $$\left|\wr\varphi + r\wr[s] - \wr\varphi' + r\wr[s]\right| = \left|(\varphi[s] + r) - (\varphi'[s] + r)\right| < \varepsilon$$

  (b) If $\varphi'[s] > \neg r$ then $\varphi'[s] \geq (\neg r + \varepsilon)$ and since $\varphi[s] > \varphi'[s] + \varepsilon$, we have $\varphi[s] < \neg r$ and thus:

  $$\left|\wr\varphi + r\wr[s] - \wr\varphi' + r\wr[s]\right| = \left|1 - 1\right| < \varepsilon$$

  (c) If $\varphi'[s] = \neg r$ then $1 + \varepsilon > (\varphi[s] + r) > 1 - \varepsilon)$ and thus $1 \geq \wr\varphi + r\wr[s] > 1 - \varepsilon$, since $\wr\varphi' + r\wr[s]$ equates to 1, the proof is complete for this case.

  2. For case C we have:

  $$\left|\varphi[s] - \varphi'[s]\right| < \frac{\varepsilon}{2} \Rightarrow \left|\wr\varphi + r\wr[s] - \wr\varphi' + r\wr[s]\right| < \frac{\varepsilon}{2}$$

  because:

  (a) If $\varphi'[s] < \neg r$ then $\varphi[s] < \neg r$ and thus:

  $$\left|\wr\varphi + r\wr[s] - \wr\varphi' + r\wr[s]\right| = \left|(\varphi[s] + r) - (\varphi'[s] + r)\right| < \frac{\varepsilon}{2}$$

  (b) If $\varphi'[s] > \neg r$ then $\varphi[s] > \neg r$ and thus:

  $$\left|\wr\varphi + r\wr[s] - \wr\varphi' + r\wr[s]\right| = \left|1 - 1\right| < \frac{\varepsilon}{2}$$

- For $\mathrm{EX}\,\varphi$ we have:

  1. For cases A and B we have:

  $$\forall s \cdot \left|\varphi[s] - \varphi'[s]\right| < \varepsilon \Rightarrow \forall s \cdot \left|(\mathrm{EX}\,\varphi)[s] - (\mathrm{EX}\,\varphi')[s]\right| < \varepsilon$$

  2. For case C we have:

  $$\forall s \cdot \left|\varphi[s] - \varphi'[s]\right| < \frac{\varepsilon}{2} \Rightarrow \forall s \cdot \left|(\mathrm{EX}\,\varphi)[s] - (\mathrm{EX}\,\varphi')[s]\right| < \frac{\varepsilon}{2}$$

  because:

  $$(\mathrm{EX}\,\varphi)[s] = \bigsqcup_{t \in S} \Big(\mathrm{R}(s, t) \sqcap \varphi[t]\Big)$$

  $$(\mathrm{EX}\,\varphi')[s] = \bigsqcup_{t \in S} \Big(\mathrm{R}(s, t) \sqcap \varphi'[t]\Big)$$

  For each node t, the distance of $\varphi[t]$ to $\varphi'[t]$ is less than ε for both cases A and B of theorem, and in case C it is less than $\frac{\varepsilon}{2}$. According to the definition of model M_ε, distance of $\mathrm{R}(s, t)$ to $\mathrm{R}'(s, t)$ is also less than ε for cases A and B, while in case C it is less than $\frac{\varepsilon}{2}$. According to properties of operators $\sqcap$ and $\sqcup$ discussed in early parts of the proof, correctness of this part can be investigated. Similarly the proof can be repeated for $\mathrm{AX}\,\varphi$.

- For $\mathrm{E}(\varphi \,\mathrm{U}\, \psi)$ we have:

1. For cases A and B we have:

$$\forall s \cdot \left|\varphi[s] - \varphi'[s]\right| < \varepsilon, \quad \forall s \cdot \left|\psi[s] - \psi'[s]\right| < \varepsilon \qquad \Rightarrow \forall s \cdot \left|\mathrm{E}(\varphi \,\mathrm{U}\, \psi)[s] - \mathrm{E}(\varphi' \,\mathrm{U}\, \psi')[s]\right| < \varepsilon$$

2. For case C we have:

$$\forall s \cdot \left|\varphi[s] - \varphi'[s]\right| < \frac{\varepsilon}{2}, \quad \forall s \cdot \left|\psi[s] - \psi'[s]\right| < \frac{\varepsilon}{2} \qquad \Rightarrow \forall s \cdot \left|\mathrm{E}(\varphi \,\mathrm{U}\, \psi)[s] - \mathrm{E}(\varphi' \,\mathrm{U}\, \psi')[s]\right| < \frac{\varepsilon}{2}$$

For the proof, we use the fixed-points concept from [Tar55]. There is at least a positive number t where $\mathrm{E}(\varphi \,\mathrm{U}\, \psi)[s] = Z_t$ and Z_i is obtained from the following recursive relation:

$$Z_i = \begin{cases} 0 & i = 0 \\ \psi[s] \sqcup (\varphi[s] \sqcap Z_{i-1}) & i \neq 0 \end{cases}$$

There is also at least a positive number t' where $\mathrm{E}(\varphi \,\mathrm{U}\, \psi)[s] = Z'_t$ and Z'_i is obtained from the following recursive relation:

$$Z'_i = \begin{cases} 0 & i = 0 \\ \psi'[s] \sqcup (\varphi'[s] \sqcap Z'_{i-1}) & i \neq 0 \end{cases}$$

Assuming $T = \max(t, t')$, we will have: $Z_t = Z_T$ and $Z'_{t'} = Z'_T$. If the distance between Z_{i-1} and Z'_{i-1} is less than ε for cases A and B, while in case C it is less than $\frac{\varepsilon}{2}$, since in cases A and B the distance $\varphi[s]$ (and $\psi[s]$) to $\varphi'[s]$ (and $\psi'[s]$) is less than ε and since it is less than $\frac{\varepsilon}{2}$ for case C, Z_i and Z'_i preserve the distance less than ε for cases A and B and $\frac{\varepsilon}{2}$ for case C. Z_0 and Z'_0 are equal, thus Z_T and Z'_T have a distance less than ε for cases A and B and $\frac{\varepsilon}{2}$ for case C. Similarly, this discussion can be repeated for $\mathrm{A}(\varphi \,\mathrm{U}\, \psi)$. With this introduction, we now explain the proof for the main title of theorem. According to the definition:

$$\mathbb{P}(M \vDash \varphi) \overset{\text{def}}{=\!=} \prod_{s \in S} \Big(\mathrm{I}(s) \to \mathbb{P}(M, s \vDash \varphi)\Big)$$
$$\mathbb{P}(M_\varepsilon \vDash \varphi_\varepsilon) \overset{\text{def}}{=\!=} \prod_{s \in S} \Big(\mathrm{I}'(s) \to \mathbb{P}(M_\varepsilon, s \vDash \varphi_\varepsilon)\Big)$$

Since for each state such as s we have the following inequality in place for cases A and B and the distance between $\mathrm{I}(s)$ and $\mathrm{I}'(s)$ is less than ε.

$$\left|\mathbb{P}(M, s \vDash \varphi) - \mathbb{P}(M_\varepsilon, s \vDash \varphi_\varepsilon)\right| < \varepsilon$$

In case C of the theorem the following inequality is in place for each state such as s; meanwhile the distance between $\mathrm{I}(s)$ and $\mathrm{I}'(s)$ is less than $\frac{\varepsilon}{2}$.

$$\left|\mathbb{P}(M, s \vDash \varphi) - \mathbb{P}(M_\varepsilon, s \vDash \varphi_\varepsilon)\right| < \frac{\varepsilon}{2}$$

Considering the properties of $\sqcap$ and $\to$ introduced in early parts of this proof, the proof is complete. □

Example 3 For K_2 model as shown Fig. 3, the following equations hold. It is shown that the error of obtained results is less than 0.25 according to K_1 model depicted in Fig. 2.

$$\mathbb{P}(K_2 \vDash \mathrm{EF}(x)) = 0.5$$
$$\mathbb{P}(K_2 \vDash \mathrm{EF}(\neg x)) = 0.75$$

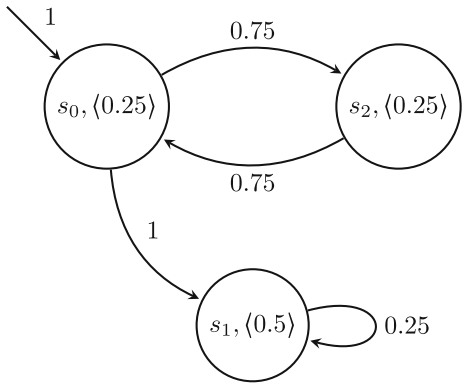


Fig. 3. Using normal approximation function with precision 0.5, K_2 is generated by approximating K_1 depicted in Fig. 2

6. Fuzzy Kripke abstraction and abstracted model approximation

Among the applications of model abstraction, the most important one is to deal with the state-space explosion by reducing the number of states (i.e., nodes), hence to convert infinite models to equivalent finite models. It is possible to deliver an abstraction for Fuzzy Kripke as it was also possible for other types of Kripke model (e.g. conventional and multi-valued Kripke models). In this section, initially *bisimulation equivalence* will be discussed and then related theorems will be presented and proved; finally, some related topics will be studied through the combination of bisimulation and approximation.

6.1. Bisimulation equivalence

Two single-source models $M = (S, X, \mathrm{R}, \mathrm{L}, s_0)$ and $M' = (S', X', \mathrm{R}', \mathrm{L}', s'_0)$ are bisimilar and relation $\sigma \in S \leftrightarrow S'$ is called bisimulation iff:

1. $(s_0, s'_0) \in \sigma$
2. $(s, s') \in \sigma \Rightarrow \mathrm{L}(s) = \mathrm{L}'(s')$
   $\wedge \forall r \in [0, 1], \forall t \in S \cdot \mathrm{R}(s, t) = r \cdot \exists (t') \in S' \cdot \mathrm{R}(s', t') = r \wedge (t, t') \in \sigma$
   $\wedge \forall r \in [0, 1], \forall t' \in S' \cdot \mathrm{R}(s', t') = r \cdot \exists (t) \in S \cdot \mathrm{R}(s, t) = r \wedge (t, t') \in \sigma$

If such a relation exists for the pair of models, the models will be called bisimilar and this similarity will be denoted as $M \sim M'$. It is easy to prove this is an equivalence relation. According to the above-mentioned definitions the following lemma and theorem can be proved.

Lemma 1 *If $M \sim M'$ for each proposition φ we have:*

$$\mathbb{P}(M \vDash \varphi) = \mathbb{P}(M' \vDash \varphi)$$

Proof. Baier and Keaton [BK] proved two similar conventional Kripke models accept any CTL* proposition with the same value; here we will use a similar framework for this proof.

- First, it is evident there must be an equivalent path π' on M' for each finite (or infinite) path π on M, such that:
  1. Each state in π has an equivalent state in π'.
  2. Transition possibility between any two consecutive states in π is equal to the transition possibility between their corresponding states in π'.
- Next, by using induction on sub-formulas we must show that:
  1. Each state proposition φ has the same value in a pair of corresponding states.
  2. Each path proposition Φ has the same value in a pair of corresponding paths.

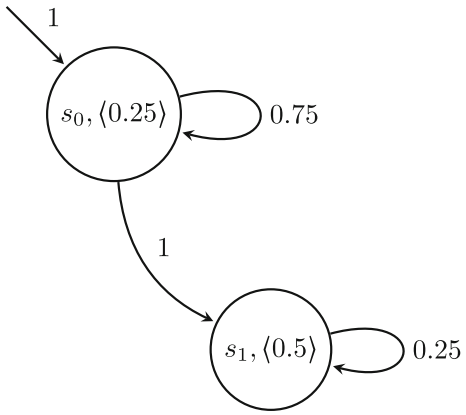


Fig. 4. Fuzzy Kripke K 3 is obtained by abstracting K 2 as shown in Fig. 3

Consequently, if both bisimilar models are single-source models, their initial states are equivalent and the equality in lemmas title holds, otherwise it is enough to modify both models into single-source models and change the verified proposition accordingly. □

Theorem 2 *If $M \sim M'$ then $M_\varepsilon \sim M'_\varepsilon$ as long as both approximated models are generated by applying the same approximation function $\tau_\varepsilon(x)$.*

Proof. If σ is a bisimulation relation between $M = (S, X, \mathrm{R}, \mathrm{L}, s_0)$ and $M' = (S', X, \mathrm{R}', \mathrm{L}', s'_0)$, it is enough to show that σ also exists for M_ε and M'_ε. It is obvious that $(s_0, s'_0) \in \sigma$; then if $(s, s') \in \sigma$ we have:

1. $\mathrm{L}(s) = \mathrm{L}'(s) \Rightarrow \mathrm{L}_\varepsilon(s) = \mathrm{L}'_\varepsilon(s)$
2. $\forall a \in [0,1], \forall t \in S, \mathrm{R}_\varepsilon(s,t) = a \cdot \exists(r) \in [0,1], \tau_\varepsilon(r) = a \wedge \mathrm{R}(s,t) = r$
   $\wedge \forall r \in [0,1], \forall t \in S, \mathrm{R}(s,t) = r \cdot \exists(t') \in S' \cdot \mathrm{R}'(s',t') = r \wedge (t,t') \in \sigma$
   $\Rightarrow \forall a \in [0,1], \forall t \in S, \mathrm{R}_\varepsilon(s,t) = a \cdot \exists(t') \in S' \cdot \mathrm{R}'_\varepsilon(s',t') = a \wedge (t,t') \in \sigma$
3. $\forall a \in [0,1], \forall t' \in S', \mathrm{R}'_\varepsilon(s',t') = a \cdot \exists(r) \in [0,1], \tau_\varepsilon(r) = a \wedge \mathrm{R}(s',t') = r$
   $\wedge \forall r \in [0,1], \forall t' \in S', \mathrm{R}(s',t') = r \cdot \exists(t) \in S \cdot \mathrm{R}(s,t) = r \wedge (t,t') \in \sigma$
   $\Rightarrow \forall a \in [0,1], \forall t' \in S', \mathrm{R}_\varepsilon(s',t') = a \cdot \exists(t) \in S \cdot \mathrm{R}_\varepsilon(s,t) = a \wedge (t,t') \in \sigma$

□

An important result of this theorem is while abstracting and approximating an infinite model to an approximated finite one, in order to avoid state-space explosion, it is possible to apply the approximation function on the infinite model then abstract resulted model to a finite one.

Example 4 For K_3 model depicted in Fig. 4, the following equation holds. These results equal the amounts calculated for Model K_2 and according to those of K_1, with errors lower than 0.25.

$$\mathbb{P}(K_2 \vDash \mathrm{EF}(x)) = 0.5$$
$$\mathbb{P}(K_2 \vDash \mathrm{EF}(\neg x)) = 0.75$$

7. Fuzzy Program Graph

By generalizing *Program Graph* model proposed in [BK] we create model $G = (S, s_0, X, \mathrm{Init}, \mathrm{Act})$. In this model, $s_0 \in S$ is the initial state, and Init is a function that defines entrance possibility to initial state. Act is a relation defining state transitions each of which has two parts; (1) a function that defines transition possibility, and (2) a function that maps possibility values from each and every attribute of the source state to those of destination state.

- $\mathrm{Init} \in \mathcal{F}_X$
- $\mathrm{Act} \in S \times S \nrightarrow \mathcal{F}_X \times \mathcal{G}_X$

where $\mathcal{F}_X \in \mathcal{P}(\text{Val(X)} \to [0, 1])$ and $\mathcal{G}_X = \mathcal{F}_X^{|X|}$ and all functions belonging to $\mathcal{F}_X$ have this constraint that they must be compatible with the following grammar's syntax.

$$\varphi ::= \text{x} \mid \kappa \mid \neg\varphi \mid \varphi \sqcap \varphi \mid \varphi \sqcup \varphi \mid \varphi \to \varphi \mid \varphi \succ \kappa \mid \varphi \prec \kappa \mid \varphi \succcurlyeq \kappa \mid \varphi \preccurlyeq \kappa \mid \varphi \approx \kappa \mid \varphi \not\approx \kappa \mid \wr \varphi \pm \kappa \wr$$
$$\kappa ::= \text{r} \mid \varphi > \kappa \mid \varphi < \kappa \mid \varphi \geq \kappa \mid \varphi \leq \kappa \mid \varphi = \kappa \mid \varphi \neq \kappa \mid \wr a\varphi \pm \kappa \wr_\varepsilon \mid \wr \theta(\kappa, \ldots, \kappa) \wr_\varepsilon$$
$$\theta \in \bigcup_{k \in \mathbb{N}^+} ([0,1]^k \to \mathbb{R}),$$
$$\varepsilon \in \mathbb{Q} \cap [0,1] \wedge \tfrac{1}{\varepsilon} \in \mathbb{N},$$
$$\text{r} \in \mathbb{Q} \cap [0,1],$$
$$a \in \mathbb{Q},$$
$$\text{x} \in \text{X}$$

where κ is a discrete valued expression and θ stands for an arbitrary analytical function with one or more arguments. Meanwhile each logical operator as well as comparison, or bounded-add/subtract, can be a specific case of θ.

7.1. Simple Fuzzy Program Graph

A special case of the FzPG in which the concrete formulas have the following syntax is called simple FzPG (SFzPG).

$$\kappa ::= \text{r} \mid \varphi > \kappa \mid \varphi < \kappa \mid \varphi \geq \kappa \mid \varphi \leq \kappa \mid \varphi = \kappa \mid \varphi \neq \kappa \mid \wr a\varphi \pm \kappa \wr_\varepsilon \mid \wr \theta(\kappa, \ldots, \kappa) \wr_\varepsilon$$

Although Act is a partial function, it can be converted to a total function with a little change. It is enough to add missing state transitions for all of nodes which are not directly connected, such that:

1. The transition possibility of the new edge is zero.
2. This edge assigns the possibility of the destination attributes by a list of zero values.

We can also make a new source by adding a dummy node $\imath$ such that Init continuously returns "1" for $\imath$, meanwhile the previous Init function must be merged with Act.

Example 5 If $X = (x, y)$, then (f, g) can be an edge of FzPG where:

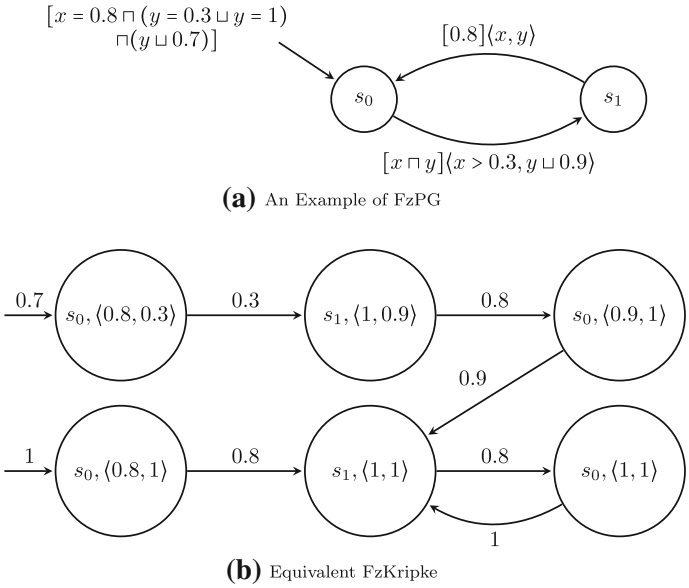
$$f(x, y) = (x > y) \sqcap (0.4 \sqcup (y < 1))$$
$$g(x, y) = \left\langle (\wr 0.3 - \wr x \wr + 2\wr x - y \wr_{0.2}) \sqcap 0.9, (x > y) \sqcup \wr \frac{\wr x \wr}{\wr x \wr + \wr y \wr + 0.1} \wr_{0.1} \right\rangle$$

7.2. Equivalency between Fuzzy Program Graph and Fuzzy Kripke

In order to express the meaning of a FzPG, an equivalent Fuzzy Kripke can be defined. There is an equivalent FzKripke $K_G = (S', X, \text{R}, \text{L}, \text{I})$ to an arbitrary FzPG like $G = (S, s_0, X, \text{Init}, \text{Act})$, such that:

1. $S' = S \times \text{Val}(X)$
2. $\forall \eta \in \text{Val}(X)$
   (a) if $s \neq s_0$, then $\forall\, s \in S \cdot \text{I}(s, \eta) = 0$
   (b) if $s = s_0$, then $\forall\, s \in S \cdot \text{I}(s, \eta) = \text{Init}(\eta)$
3. $\forall\, \eta \in \text{Val}(X), \forall\, s \in S \cdot \text{L}(s, \eta) = \eta$
4. $\forall\, \eta, \eta' \in \text{Val}(X)$
   (a) if $\text{Act}(s, s')$ is not defined, then $\forall\, s, s' \in S \cdot \text{R}((s, \eta), (s', \eta')) = 0$
   (b) if $(\text{A}, \text{B}) = \text{Act}(s, s')$, $\eta' = \text{B}(\eta)$ then $\forall\, s, s' \in S \cdot \text{R}((s, \eta), (s', \eta')) = \text{A}(\eta)$

If a finite number of members in $\text{Val}(X)$ are setting the result of Init function to positive values then there are a finite number of states for the resulted FzKripke; otherwise the number of states may be infinite for the resulted FzKripke.



(a) An Example of FzPG

(b) Equivalent FzKripke

Fig. 5. An example of FzPG alongside its equivalent FzKripke model

When the equivalent FzKripke model accepts a proposition with a certain possibility, we say that the FzPG also accepts the proposition with the same possibility. Equivalent to each FzKripke, an FzPG with the same number of states can be defined; however, this conversion is not valuable. The main goal of defining FzPG is to express FzKripke in a highly compressed format.

8. From FzPG to corresponding abstracted and approximated FzKripke

In order to check and verify the temporal properties of a particular FzPG, an equivalent finite abstract model is required. It is also possible to convert a particular FzPG to an infinite FzKripke. Whenever error rate is tolerable during evaluation of logical propositions, a finite approximated model corresponding to the infinite Kripke model is obtainable. Further theorems present a framework for constructing such finite models from an FzPG. Relevant algorithms are easy to design regarding the results of these theorems.

Theorem 3 *For each FzPG like G, a number like* $\delta = \frac{1}{N}$ *exists such that, for each formula* $f \in \mathcal{F}_X$ *used in the edges of the graph, we have:*

$$[\![f(\eta)]\!]_\delta = f([\![\eta]\!]_\delta)$$

Proof. For each discrete sub-formula κ, we consider a resolution degree such as $\rho(\kappa)$ and define it recursively. We consider the number δ the greatest possible value that satisfies the following constraints. In these relations, κ is a discrete sub-formula, φ is an arbitrary formula, $\bowtie$ represents comparison operators, and $\otimes$ represents quasi-comparison operators.

$$\kappa = r \Rightarrow \rho(\kappa) = r \tag{33}$$

$$\kappa = \wr \theta(\kappa_1, \ldots, \kappa_t) \wr_\varepsilon \Rightarrow \frac{\rho(\kappa_1)}{\delta} \in \mathbb{Z}, \ldots, \frac{\rho(\kappa_t)}{\delta} \in \mathbb{Z}, \quad \rho(\kappa) = \varepsilon \tag{34}$$

$$\kappa = \wr\, a\varphi' \pm \kappa' \wr_\varepsilon = \frac{\varepsilon}{a\delta} \in \mathbb{Z}, \quad \frac{\rho(\kappa')}{a\delta} \in \mathbb{Z}, \rho(\kappa) = \varepsilon \tag{35}$$

$$\kappa = \varphi' \bowtie \kappa' \Rightarrow \frac{\rho(\kappa')}{\delta} \in \mathbb{Z}, \quad \rho(\kappa) = 1 \tag{36}$$
$$(\bowtie \quad \in \{\geq, \leq, >, <, =, \neq\})$$

$$\varphi = \wr \varphi' \pm \kappa \wr_\varepsilon \text{ or } \varphi = \varphi' \otimes \kappa \text{ or } \varphi = \kappa \Rightarrow \frac{\rho(\kappa)}{\delta} \in \mathbb{Z} \tag{37}$$
$$(\otimes \in \{\succcurlyeq \preccurlyeq \succ \prec \approx \not\approx\})$$

There is surely a number like δ that satisfies these constraints. Also, any number δ' by which δ is divisible, satisfies the above constraints and can be considered as another answer of the theorem. For discrete sub-formulas, proposition $\frac{\kappa}{\rho(\kappa)} \in \mathbb{Z}$ can be investigated.

Now we must show for each sub-formula φ included in the Formula f, the property $[\![\varphi(\eta)]\!]_\delta = \varphi([\![\eta]\!]_\delta)$ holds, because:

$$\varphi = x \Rightarrow [\![x(\eta)]\!]_\delta = x([\![\eta]\!]_\delta) \tag{38}$$

$$\varphi = \neg\varphi' \Rightarrow [\![\varphi(\eta)]\!]_\delta = 1 - [\![\varphi(\eta)]\!]_\delta = 1 - \varphi([\![\eta]\!]_\delta) = \varphi'([\![\eta]\!]_\delta) \tag{39}$$

$$\varphi = \varphi_1 \sqcap \varphi_2 \Rightarrow [\![\varphi(\eta)]\!]_\delta = [\![\varphi_1(\eta)]\!]_\delta \sqcap [\![\varphi_2(\eta)]\!]_\delta = \varphi_1([\![\eta]\!]_\delta) \sqcap \varphi_2([\![\eta]\!]_\delta) = \varphi([\![\eta]\!]_\delta) \tag{40}$$

Proposition $[\![\varphi \sqcap \psi]\!]_\delta = [\![\varphi]\!]_\delta \sqcap [\![\psi]\!]_\delta$ also holds. In order to prove this proposition, when $[\![\varphi]\!]_\delta < [\![\psi]\!]_\delta$ then $\varphi < \psi$ and the equality is easily verifiable. Therefore the theorem is also established for the $>$ operator. When $[\![\varphi]\!]_\delta < [\![\psi]\!]_\delta$, both φ and ψ are divisible by δ or neither of them are divisible by it; in the latter case the following proof is established and can be repeated for logical operators like $\rightarrow$ and $\sqcup$.

$$\varphi = (p + \alpha)\delta, \quad \psi = (p + \beta)\delta$$
$$\Rightarrow \varphi \sqcap \psi = \big(p + \min(\alpha, \beta)\big)\delta, \quad [\![\varphi \sqcap \psi]\!]_\delta = [\![\varphi]\!]_\delta \sqcap [\![\psi]\!]_\delta = [\![\varphi]\!]_\delta = [\![\psi]\!]_\delta$$
$$p \in \mathbb{Z}, \quad \alpha, \beta \in (0, 1)$$

In case $\varphi = \kappa$, If κ is a constant number such as r, and r is divisible by δ, the following relation holds because it is independent on η.

$$\varphi = \kappa \Rightarrow [\![\varphi(\eta)]\!]_\delta = \varphi(\eta) = \varphi([\![\eta]\!]_\delta) \tag{41}$$

On the other hand when $\varphi(\eta) = \wr \theta\big(\kappa_1(\eta), \ldots, \kappa_t(\eta)\big) \wr_\varepsilon$, since $[\![\wr \alpha \wr_\varepsilon]\!]_\delta = \wr \alpha \wr_\varepsilon$ and above property holds for each value of κ_i, property also holds for the entire formula φ.

Regarding the equation $\varphi(\eta) = \wr\, a\varphi'(\eta) \pm \kappa'(\eta) \wr_\varepsilon$ we have:

$$[\![\varphi(\eta)]\!]_\delta = \varphi(\eta) = \wr\, a\varphi'(\eta) \pm \kappa'(\eta) \wr_\varepsilon = \wr\, a[\![\varphi'(\eta)]\!]_\delta \pm \kappa'(\eta) \wr_\varepsilon = \wr\, a\varphi'([\![\eta]\!]_\delta) \pm \kappa'([\![\eta]\!]_\delta) \wr_\varepsilon = \varphi([\![\eta]\!]_\delta) \tag{42}$$

Some parts of above equations must be proved.

$$\wr\, a\varphi'(\eta) \pm \kappa'(\eta) \wr_\varepsilon = \wr\, a[\![\varphi'(\eta)]\!]_\delta \pm \kappa'(\eta) \wr_\varepsilon \tag{43}$$

If $\varphi'(\eta)$ is dividable by δ, it is straight forward to evaluate this relation, otherwise since $\frac{\rho(\kappa')}{a\delta} \in \mathbb{Z}$ and $\frac{\kappa'}{\rho(\kappa')} \in \mathbb{Z}$ we conclude that: $\kappa'(\eta)$ is divisible by $a\delta$, and $\frac{\varepsilon}{a\delta} \in \mathbb{Z}$.

$$\varphi'(\eta) = (p + \alpha)\delta, \kappa'(\eta) = qa\delta, \varepsilon = ea\delta$$
$$\wr a\varphi'(\eta) \pm \kappa'(\eta)\int_{\varepsilon} = \varepsilon \left\lfloor \frac{a}{\varepsilon}\varphi'(\eta) \pm \frac{\kappa'(\eta)}{\varepsilon} \right\rfloor = \varepsilon \left\lfloor \frac{p \pm q + \alpha}{e} \right\rfloor = \varepsilon \left\lfloor \frac{p \pm q}{e} \right\rfloor$$
$$[\![\varphi'(\eta)]\!]_\delta = (p + 0.5)\delta$$
$$\wr a[\![\varphi'(\eta)]\!]_\delta \pm \kappa'(\eta)\int_{\varepsilon} = \varepsilon \left\lfloor \frac{p \pm q + 0.5}{e} \right\rfloor = \varepsilon \left\lfloor \frac{p \pm q}{e} \right\rfloor$$
$$p, q, e \in \mathbb{Z}, \quad \alpha \in (0, 1)$$

If $\varphi = \varphi' \bowtie \kappa'$, where $\bowtie$ is a comparison operator, then we must prove the following condition:

$$[\![\kappa'(\eta)]\!]_\delta = \kappa'(\eta) = \kappa'([\![\eta]\!]_\delta), [\![\varphi'(\eta)]\!]_\delta = \varphi'([\![\eta]\!]_\delta) \Rightarrow [\![\varphi(\eta)]\!]_\delta = \varphi(\eta) = \varphi([\![\eta]\!]_\delta) \tag{44}$$

Proof is as follows:

$$\varphi(\eta) = \varphi'(\eta) \bowtie \kappa'(\eta) = [\![\varphi'(\eta)]\!]_\delta \bowtie \kappa'(\eta) = \varphi'([\![\eta]\!]_\delta) \bowtie \kappa'([\![\eta]\!]_\delta) = \varphi([\![\eta]\!]_\delta)$$

Relation $\varphi'(\eta) \bowtie \kappa'(\eta) = [\![\varphi'(\eta)]\!]_\delta \bowtie \kappa'(\eta)$ is correct because $\kappa'(\eta)$ is divisible by δ. If $\varphi'(\eta)$ is also divisible by δ, checking of the equation is simple, otherwise we have:

$$\varphi'(\eta) = (p + \alpha)\delta, \kappa'(\eta) = q\delta$$
$$\varphi'(\eta) \bowtie \kappa'(\eta) = (p + \alpha \bowtie q) = (p + 0.5 \bowtie q) = [\![\varphi'(\eta)]\!]_\delta \bowtie \kappa'(\eta)$$
$$p, q \in \mathbb{Z}, \quad \alpha \in (0, 1)$$

It is easy to check the property whenever $\bowtie$ equates to $=$ or $\neq$. If $\bowtie$ equates to $\geq$ then p is greater than or equal to q, by adding 0.5 to p comparison will be converted to $>$. Other cases can be checked, similarly.

$$\varphi = \wr\varphi' \pm \kappa\int \Rightarrow [\![\varphi(\eta)]\!]_\delta = \wr[\![\varphi'(\eta)]\!]_\delta \pm \kappa(\eta)\int = \wr\varphi'([\![\eta]\!]_\delta) \pm \kappa([\![\eta]\!]_\delta)\int = \varphi([\![\eta]\!]_\delta)$$

Above proposition is correct, because $\kappa(\eta)$ is divisible by δ and $\kappa(\eta) = \kappa([\![\eta]\!]_\delta)$. Similar to bounded-add/subtract in the form $\wr\varphi' \pm \kappa\int$, for quasi-comparison operators the theorem is established and can be verified.

Hereafter in this paper, δ_G represents the largest δ for graph G. □

Theorem 4 *Model K is equivalent infinite FzKripke to a particular FzPG such as G. K_G can be approximated to K_δ using normal-saturation function (with precision δ which is a divisor of δ_G). This new model can be abstracted to a finite model such as K'_G with $|S|(\frac{2}{\delta} + 1)^{|X|}$ states, and for each formula φ (that can be approximated to φ_δ) we have:*

$$\left| \mathbb{P}(K_G \vDash \varphi) - \mathbb{P}(K'_G \vDash \varphi_\delta) \right| < \frac{\delta}{2}$$

Proof. Assume that $G = (S, s_0, X, \text{Init}, \text{Act})$ is a particular FzPG and $K_G = (S_1, X, R_1, L_1, \iota)$ as its equivalent single-source FzKripke; thus $K_\delta = (S_2, X, R_2, L_2, \iota)$ is also a single-source FzKripke. Also assume $\text{Act}(s, t) = (A_{st}, B_{st})$. We build $K'_G = (S_3, X, R_3, L_3, \iota)$ as follows:

$$S_3 = S \times \left\{ q\frac{\delta}{2} \,\middle|\, q \in \left\{0, \ldots, \frac{\delta}{2}\right\} \right\}^{|X|} \cup \{\iota\}$$
$$L_2(\langle s, \mu\rangle) = \mu$$
$$R_3(\iota, \langle s_0, \mu\rangle) = \text{Init}(\mu)$$
$$\text{if } s \neq s_0 \text{ then } R_3(\iota, \langle s_i, \mu\rangle) = 0$$
$$R_3(\langle s, \mu\rangle, \langle t, B_{st}\rangle) = A_{st}$$
$$\text{if } \mu' \neq B_{st} \text{ then } R_3(\langle s, \mu\rangle, \langle t, \mu'\rangle) = 0$$

We show K_δ and K'_G are bisimilar according to the following relation:

$$\sigma = \Big\{\big(\langle s,\eta\rangle, \langle s, [\![\eta]\!]_\delta\rangle | \langle s,\eta\rangle \in S_2 - \{\iota\}\big)\Big\} \cup \big\{(\iota,\iota)\big\}$$

If $R_1\big(\langle s,\eta\rangle, \langle t, B_{st}(\eta)\rangle\big) = A_{st}(\eta)$ then we have:

$$R_2(\langle s,\eta\rangle, \langle t, B_{st}(\eta)\rangle) = [\![A_{st}(\eta)]\!]_\delta = A_{st}([\![\eta]\!]_\delta),\ L_2(\langle s,\eta\rangle) = [\![\eta]\!]_\delta,\ L_2(\langle t, B_{st}(\eta)\rangle) = [\![B_{st}(\eta)]\!]_\delta = B_{st}([\![\eta]\!]_\delta)$$
$$R_3(\langle s, [\![\eta]\!]_\delta\rangle, \langle t, B_{st}([\![\eta]\!]_\delta)\rangle) = A_{st}([\![\eta]\!]_\delta),\ L_3(\langle s, [\![\eta]\!]_\delta\rangle) = [\![\eta]\!]_\delta,\ L_3(\langle t, B_{st}([\![\eta]\!]_\delta)\rangle) = B_{st}([\![\eta]\!]_\delta)$$

And if $R_1(\iota, \langle s_0,\eta\rangle) = \mathrm{Init}(\eta)$ we have:

$$R_2(\iota, \langle s_0,\eta\rangle) = [\![\mathrm{Init}(\eta)]\!]_\delta = \mathrm{Init}([\![\eta]\!]_\delta)$$
$$R_3(\iota, \langle s_0, [\![\eta]\!]_\delta\rangle) = \mathrm{Init}([\![\eta]\!]_\delta)$$

We now show that σ has bisimulation properties.

$$L_2(\langle s,\eta\rangle) = L_3(\langle s, [\![\eta]\!]_\delta\rangle), \forall\, r \in [0,1] \cdot R_2(\langle s,\eta\rangle, \langle t,\eta'\rangle) = r \Rightarrow R_3(\langle s, [\![\eta]\!]_\delta\rangle, \langle t, [\![\eta']\!]_\delta\rangle) = r$$

Whenever $\eta' = B_{st}(\eta)$ then $r = A_{st}([\![\eta]\!]_\delta)$ and obviously $[\![\eta']\!]_\delta = B_{st}([\![\eta]\!]_\delta)$, otherwise $r = 0$; also we have:

$$\forall r \in [0,1] \cdot R_2(\iota, \langle s,\eta\rangle) = r \Rightarrow R_3(\iota, \langle s, [\![\eta]\!]_\delta\rangle) = r$$

Whenever $s = s_0$ then $r = \mathrm{Init}([\![\eta]\!]_\delta)$ otherwise r equals zero.

$$\forall r \in [0,1] \cdot R_3(\langle s,\mu\rangle, \langle t,\mu'\rangle) = r \Rightarrow R_2(\langle s,\mu\rangle, \langle t,\mu'\rangle) = r$$

According to definition of normal-saturation function, $\mu = [\![\mu]\!]_\delta$ and $\mu' = [\![\mu']\!]_\delta$. Now if $\mu' = B_{st}(\mu)$ then $r = A_{st}(\mu)$ otherwise r equals zero. Similarly we have:

$$\forall r \in [0,1] \cdot R_3(\iota, \langle s,\mu\rangle) = r \Rightarrow R_2(\iota, \langle s,\mu\rangle) = r$$

□

However, K'_G is a finite model and the error rate of propositions' values is less than $\frac{\delta}{2}$, since δ is small, the number of states is still large. Albeit the proposition values will have higher error rates, but the new model can be re-approximated with precision Δ (which is less precise in comparison with δ). The question is whether we can approximate FzPG and simultaneously create an approximated FzKripke model with minimal states before forming the approximated K'_G. If it is possible, regarding the compactness of FzPG, the approximation cost in terms of space and time will be reduced; meanwhile the final approximated FzKripke will be generated directly. To answer this question the following definitions and theorems, are presented.

8.1. Normal Δ-approximable formula

Formula φ from $\mathcal{F}_X$ is called normal Δ-approximable, if by applying the approximation τ_Δ (which will be defined later) on it, the new formula and its sub-formulas satisfy the constraints mentioned in the proof of Theorem 3 [relations (33)–(38) for $\delta = \Delta$]. Approximation τ_Δ for each formula φ according to definition

of $\mathcal{F}_X$ is as follows:

$$\begin{aligned}
&\varphi = x \Rightarrow \tau_\Delta(\varphi) = x\\
&\varphi = \neg\varphi' \Rightarrow \tau_\Delta(\varphi) = \neg\tau_\Delta(\varphi')\\
&\varphi = \varphi' \star \varphi'' \Rightarrow \tau_\Delta(\varphi) = \tau_\Delta(\varphi') \star \tau_\Delta(\varphi'')\\
&\qquad (\star \in \{\sqcap, \sqcup, \rightarrow\})\\
&\varphi = \wr \varphi' \pm \kappa \int \Rightarrow \tau_\Delta(\varphi) = \wr \tau_\Delta(\varphi') \pm \kappa \int\\
&\varphi = \varphi' \otimes \kappa \Rightarrow \tau(\varphi) = \tau_\Delta(\varphi') \otimes \kappa\\
&\qquad (\otimes \in \{\gtrdot, \preccurlyeq, \succ, \prec, \approx, \not\approx\})\\
&\varphi = r \Rightarrow \tau_\Delta(\varphi) = [\![r]\!]_\Delta\\
&\qquad (r \text{ is const.})\\
&\varphi = \varphi' \bowtie \kappa' \Rightarrow \tau_\Delta(\varphi) = \tau_\Delta(\varphi') \bowtie \kappa'\\
&\qquad (\bowtie \in \{\geq, \leq, >, <, =, \neq\})\\
&\varphi = \wr a\varphi' \pm \kappa' \int_\varepsilon \Rightarrow \tau_\Delta(\varphi) = \wr a\tau_\Delta(\varphi') \pm \kappa' \int_\varepsilon\\
&\varphi = \wr \theta(\kappa_1, \ldots, \kappa_t) \int_\varepsilon \Rightarrow \tau_\Delta(\varphi) = \varphi
\end{aligned}$$

Notice, in this section we use φ_Δ as the abbreviation of $\tau_\Delta(\varphi)$.

Example 6

$$\varphi = \big((x \sqcap 0.24 \sqcap \wr y + 0.7 \int) \sqcup 0.815\big) > 0.4 \Rightarrow \varphi_{0.1} = \big((x \sqcap 0.25 \sqcap \wr y + 0.7 \int) \sqcup 0.85\big) > 0.4$$

Definition 1 The FzPG G is called normal Δ-precision approximable if all formulas involved in transition edges or its initial function are Δ-precision approximable. The approximated FzPG is called G_Δ.

Lemma 2 *For each formula φ, involved in the FzPG G, which is normal Δ-precision approximable we have:*

$$[\![\varphi(\eta)]\!]_\Delta = [\![\varphi_\Delta(\eta)]\!]_\Delta = \varphi_\Delta([\![\eta]\!]_\Delta)$$

Proof. Using induction on sub-formulas of formula φ , we can prove the correctness of proposition $[\![\varphi(\eta)]\!]_\Delta = [\![\varphi_\Delta(\eta)]\!]_\Delta$.

$$\begin{aligned}
&\varphi(\eta) = r \Rightarrow \varphi_\Delta(\eta) = [\![r]\!]_\Delta, [\![r]\!]_\Delta = [\![[\![r]\!]_\Delta]\!]_\Delta\\
&\varphi(\eta) = x(\eta) \text{ or } \varphi(\eta) = \wr \theta(\kappa_1(\eta), \ldots, \kappa_t(\eta)) \int_\varepsilon \Rightarrow \varphi_\Delta(\eta) = \varphi(\eta) \Rightarrow [\![\varphi(\eta)]\!]_\Delta = [\![\varphi_\Delta(\eta)]\!]_\Delta\\
&\varphi(\eta) = \neg\varphi'(\eta) \Rightarrow [\![\varphi(\eta)]\!]_\Delta = \neg[\![\varphi'(\eta)]\!]_\Delta = \neg[\![\varphi'_\Delta(\eta)]\!]_\Delta = [\![\neg\varphi'_\Delta(\eta)]\!]_\Delta = [\![\varphi_\Delta(\eta)]\!]_\Delta\\
&\varphi(\eta) = \varphi'(\eta) \star \varphi''(\eta) \Rightarrow [\![\varphi(\eta)]\!]_\Delta = [\![\varphi'(\eta)]\!]_\Delta \star [\![\varphi''(\eta)]\!]_\Delta = [\![\varphi'_\Delta(\eta)]\!]_\Delta \star [\![\varphi''_\Delta(\eta)]\!]_\Delta\\
&\qquad = [\![\varphi'_\Delta(\eta) \star \varphi''_\Delta(\eta)]\!]_\Delta = [\![\varphi_\Delta(\eta)]\!]_\Delta\\
&\varphi(\eta) = \wr \varphi'(\eta) \pm \kappa(\eta) \int \Rightarrow \frac{\rho(\kappa)}{\Delta} \in \mathbb{Z}, [\![\varphi(\eta)]\!]_\Delta = [\![\wr \varphi'(\eta) \pm \kappa(\eta) \int]\!]_\Delta = \wr [\![\varphi'(\eta)]\!]_\Delta \pm \kappa(\eta) \int\\
&\qquad = \wr [\![\varphi'_\Delta(\eta)]\!]_\Delta \pm \kappa(\eta) \int = [\![\wr \varphi'_\Delta(\eta) \pm \kappa(\eta) \int]\!]_\Delta = [\![\varphi_\Delta(\eta)]\!]_\Delta
\end{aligned}$$

In case of comparison and quasi-comparison operators, proof is same as the bounded-add/subtract. If $\varphi(\eta) = \wr a\varphi'(\eta) \pm \kappa(\eta) \int_\varepsilon$ the proof is different:

$$[\![\varphi_\Delta(\eta)]\!]_\Delta = [\![\wr a\varphi'_\Delta(\eta) \pm \kappa(\eta) \int_\varepsilon]\!]_\Delta = \wr a\varphi'_\Delta(\eta) \pm \kappa(\eta) \int_\varepsilon$$

According to constraint (38) we have $\frac{\varepsilon}{\Delta} \in \mathbb{Z}$, hence above equation holds. Also we have:

$$[\![\varphi(\eta)]\!]_\Delta = [\![\wr a\varphi'(\eta) \pm \kappa(\eta) \int_\varepsilon]\!]_\Delta = \wr a\varphi'(\eta) \pm \kappa(\eta) \int_\varepsilon$$

On the other hand, according to (39) we have:

$$\wr a\varphi'_{\Delta}(\eta) \pm \kappa(\eta)\wr_{\varepsilon} = \wr a[\![\varphi'_{\Delta}(\eta)]\!]_{\Delta} \pm \kappa(\eta)\wr_{\varepsilon}$$
$$\wr a\varphi'(\eta) \pm \kappa(\eta)\wr_{\varepsilon} = \wr a[\![\varphi'(\eta)]\!]_{\Delta} \pm \kappa(\eta)\wr_{\varepsilon}$$

Because of equation $[\![\varphi'_{\Delta}(\eta)]\!]_{\Delta} = [\![\varphi'(\eta)]\!]_{\Delta}$, and considering above equations, $[\![\varphi_{\Delta}(\eta)]\!]_{\Delta} = [\![\varphi(\eta)]\!]_{\Delta}$ holds. Since these sub-formals satisfy constraints (33)–(38), the proof of $[\![\varphi_{\Delta}(\eta)]\!]_{\Delta} = \varphi_{\Delta}([\![\eta]\!]_{\Delta})$ is similar to the above proof of the Theorem 3. □

Theorem 5 *While G is a Δ-precision approximable FzPG, its equivalent infinite FzKripke K_G can be approximated to K_Δ using normal-saturation function (with precision Δ which is a divisible by δ_G). This new model can be abstracted to a finite model such as K'_G with $|S|(\frac{2}{\Delta}+1)^{|X|}$ states, and for each formula φ (that can be approximated to φ_Δ) we have:*

$$\left|\mathbb{P}(K_G \vDash \varphi) - \mathbb{P}(K'_G \vDash \varphi_\Delta)\right| < \frac{\Delta}{2}$$

Proof. The result of Approximation on $K_G\Delta$ (using normal-saturation function with precision Δ) is denoted as $K_G\Delta'$. According to the previous theorem, the recent model can be abstracted to a finite model D_Δ with $|S|(\frac{2}{\Delta}+1)^{|X|}$ nodes. For each approximable FzCTL* proposition such as φ, according to the results of the previous theorems, we have:

$$\left|\mathbb{P}(K_G \vDash \varphi) - \mathbb{P}(K_\Delta \vDash \varphi_\Delta)\right| < \frac{\Delta}{2}$$

If it can be proved that $K_\Delta \sim K'_G$, it is easy to show that:

$$\left|\mathbb{P}(K_G \vDash \varphi) - \mathbb{P}(K'_G \vDash \varphi_\Delta)\right| < \frac{\Delta}{2}$$

Assume $K_G = (S_1, X, R_1, L_1, \iota)$, $K_\Delta = (S_2, X, R_2, L_2, \iota)$ and $K'_G = (S_3, X, R_3, L_3, \iota)$ are single-source models. Also assume that $\mathrm{Act}(s,t) = (A_{st}, B_{st})$. We call normal approximation of A_{st}, B_{st} and Init (with precision Δ) A'_{st}, B'_{st} and Init′, respectively. We show that K_Δ and K'_G are bisimilar and bisimulation relation is as follows:

$$\sigma = (\langle s,\eta\rangle, \langle s, [\![\eta]\!]_\Delta\rangle : \langle s,\eta\rangle \in S_2 - \iota \cup (\iota,\iota)$$

If $R_1(\langle s,\eta\rangle, \langle t, B_{st}(\eta)\rangle) = A_{st}(\eta)$ we have:

$$R_2(\langle s,\eta\rangle, \langle t, B_{st}(\eta)\rangle) = [\![A_{st}(\eta)]\!]_\Delta = A'_{st}([\![\eta]\!]_\Delta),\ L_2(\langle s,\eta\rangle) = [\![\eta]\!]_\Delta,\ L_2(\langle t, B_{st}(\eta)\rangle) = [\![B_{st}(\eta)]\!]_\Delta = B'_{st}([\![\eta]\!]_\Delta)$$
$$R_3(\langle s, [\![\eta]\!]_\Delta\rangle, \langle t, B'_{st}([\![\eta]\!]_\Delta)\rangle) = A'_{st}([\![\eta]\!]_\Delta),\ L_3(\langle s, [\![\eta]\!]_\Delta\rangle) = [\![\eta]\!]_\Delta,\ L_3(\langle t, B'_{st}([\![\eta]\!]_\Delta)\rangle) = B'_{st}([\![\eta]\!]_\Delta)$$

And if $R_1(\iota, \langle s_0,\eta\rangle) = \mathrm{Init}(\eta)$ we have:

$$R_2(\iota, \langle s_0,\eta\rangle) = [\![\mathrm{Init}(\eta)]\!]_\Delta = \mathrm{Init}'([\![\eta]\!]_\Delta)$$
$$R_3(\iota, \langle s_0, [\![\eta]\!]_\Delta\rangle) = \mathrm{Init}'([\![\eta]\!]_\Delta)$$

Now we check whether σ has the properties of bisimulation relation or not:

$$L_2(\langle s,\eta\rangle) = L_3(\langle s, [\![\eta]\!]_\Delta\rangle), \forall\, r \in [0,1] \cdot R_2(\langle s,\eta\rangle, \langle t,\eta'\rangle) = r \Rightarrow R_3(\langle s, [\![\eta]\!]_\Delta\rangle, \langle t, [\![\eta']\!]_\Delta\rangle) = r$$

Because if $\eta' = B_{st}(\eta)$ then $r = A'_{st}([\![\eta]\!]_\Delta)$ and surely $[\![\eta']\!]_\Delta = B'_{st}([\![\eta]\!]_\Delta)$, otherwise $r = 0$; also we have:

$$\forall r \in [0,1] \cdot R_2(\iota, \langle s,\eta\rangle) = r \Rightarrow R_3(\iota, \langle s, [\![\eta]\!]_\Delta\rangle) = r$$

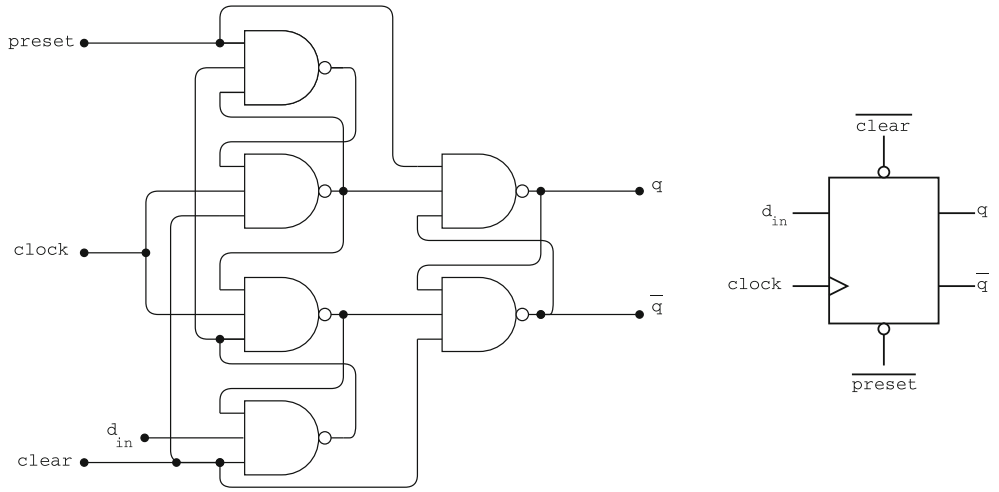
Because if $s = s_0$ then $r = \mathrm{Init}'([\![\eta]\!]_\Delta)$ otherwise r is zero.

$$\forall r \in [0,1] \cdot R_3(\langle s,\mu\rangle, \langle t,\mu'\rangle) = r \Rightarrow R_2(\langle s,\mu\rangle, \langle t,\mu'\rangle) = r$$

According to the definition of normal-saturation function, $\mu = [\![\mu]\!]_\Delta$ and $\mu' = [\![\mu']\!]_\Delta$. Now if $\mu' = B'_{st}(\mu)$ then $r = A'_{st}(\mu)$, otherwise r is zero. Similarly:

$$\forall r \in [0,1] \cdot R_3(\iota, \langle s,\mu\rangle) = r \Rightarrow R_2(\iota, \langle s,\mu\rangle) = r$$

□



(a) Circuit Diagram **(b)** Block Diagram

Fig. 6. Quaternary D-flip flop

9. Case study

Most of recent studies in digital circuit design showed a vast amount of interest in model checking and generally formal verification techniques to verify synchronous and asynchronous sequential logics. In particular, model checking is used to find bugs such as lack of stability, and hazard occurrence early in the design phase of a digital circuit. Burch et al. [BCL+94] introduced a symbolic model checking approach to verify sequential circuits by modeling sequential behavior of a circuit using *Binary Decision Diagrams* (BDD's) and then using CTL for model verification.

Synchronous sequential circuits are typically easy to check, asynchronous sequential circuits are also straightforward to verify whenever all logic gates used in an asynchronous circuit have equal propagation delays. However circuits with special design models such as fundamental-mode, speed-independent, and delay-intensive are also easy to verify, asynchronous circuits whose components have different or even non-deterministic propagation delays are difficult to verify [Cun04], yet Maler and Pnueli [MP95] used timed automata to verify this class of asynchronous circuits.

Most recently in addition to the binary logic circuits researchers are also focused on fuzzy logic, and even multi-valued logic circuits along with their applications [EFR74, Smi81, Hur84]. Much of the work in this area includes designing fuzzy logic gates, fuzzy flip-flops, and their combination in larger circuits (such as memory units) [KHH87, Smi88].

Hirota et al. proposed the concept of fuzzy flip-flops while they presented the idea for designing fuzzy hardware systems using these flip-flops [HO89, OHK+95, OHK96, HP95, HP95]. Fuzzy flip-flops are made of fuzzy logic gates through having some extensions to binary logic flip-flops. So far, no specific study has been made to verify multi-valued logic circuits. In this section we used FzPG to model the multi-valued D flip-flop presented in [Gur10] (depicted in Fig. 6), afterward we used FzCTL to verify the model and discover design errors. The provided technique in this section can be generalized for other asynchronous multi-valued logic circuits. For further readability of the paper we use the following abbreviations while examining this circuit:

- C : *clock*
- R : *clear*
- Q : q
- P : *preset*
- D : d_{in}
- $\bar{Q}$: $\bar{q}$

The domain of ϑ in Table 2 is $\{0, \frac{1}{3}, \frac{2}{3}, 1\}$ for quaternary logic; similarly, the domain of a (k+1)-valued logic is $\{0, \frac{1}{k}, \frac{2}{k}, \ldots, 1\}$. For simplicity, we assume propagation delays for all logic gates are equal to $\Delta = 2^{-h}$ time units where h is a positive integer. We also assume that P, R, Q values are stable, but C is a periodic pulse as depicted in Fig. 7

Table 2. Quaternary D-flip flop truth table

Inputs				Outputs	
P	R	D	C	Q	$\bar{Q}$
0	1	X	X	1	0
1	0	X	X	0	1
1	1	ϑ	$\uparrow$	ϑ	$\neg\vartheta$
0	0	X	X	?	?

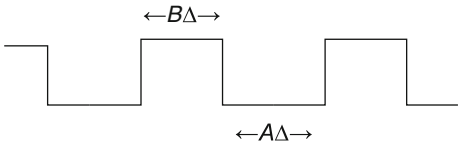


Fig. 7. The periodic clock pulse where A, and B are positive integers

Considering the periodic clock pulse shown in Fig. 7, larger values of A and B does not affect circuit's behavior but slowing it down. On the other hand, it is evident if values of A and B are smaller than a certain threshold there is no time for stabilizing the output of the circuit; thus its operation will be inaccurate. In a multi-valued logic flip-flop the minimum value for both A and B is equal to 3 (inaccurate circuit operation while using values less than 3 can be investigated by model checking methods) while we consider $N = \frac{1}{\Delta}$ to be the maximum value for A and B (the maximum value may exceed the provided N without affecting the circuit except slowing it down).

We labeled outputs of all gates in Fig. 6; (i.e., we labeled four most left placed gates in the diagram from top to bottom as x, y, z, and w respectively). We also used the prime notation to label the outputs after Δ time units. The output for each gate can be calculated easily as follows:

$$x' = \text{Nand}(P, w, y)$$
$$y' = \text{Nand}(x, C, R)$$
$$z' = \text{Nand}(y, C, w)$$
$$w' = \text{Nand}(z, R, D)$$
$$Q' = \text{Nand}(P, y, \bar{Q})$$
$$\bar{Q}' = \text{Nand}(Q, z, R)$$

The Nand gate is simply defined as follows:

$$\text{Nand}(a, b, c) = \neg(a \sqcap b \sqcap c)$$

In this stage we define a FzPG called G with two states s_1, s_2, please see Fig. 8. In s_1, C equates to "0" while in s_2 it equates to "1". The set of attributes is as follows:

$$X = \langle T, u, P, R, D, C, x, y, z, w, Q, \bar{Q} \rangle$$

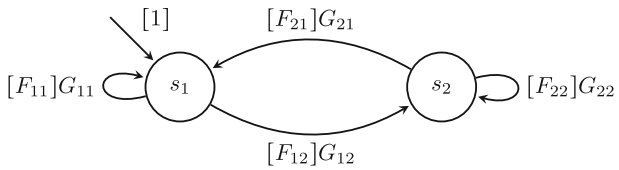


Fig. 8. The corresponding FzPG to the multi-valued flip-flop shown in Fig. 6

Details of participant functions in the FzPG are shown below:

$$I = (T = 0) \sqcap (C = 0) \sqcap (u = 0)$$
$$F_{11} = F_{22} = (T < 1)$$
$$F_{12} = F_{21} = (T \geq 3\Delta)$$
$$G_{12} = \langle 0, 1, P, R, D, 1, x, y, z, w, Q, \bar{Q} \rangle$$
$$G_{21} = \langle 0, u, P, R, D, 0, x, y, z, w, Q, \bar{Q} \rangle$$
$$G_{11} = G_{22} = \langle \wr T + \Delta \int, u, P, R, D, C, \ \text{Nand}(P, w, y), \text{Nand}(x, C, R), \text{Nand}(y, C, w), \text{Nand}(z, R, D), \text{Nand}(P, y, \bar{Q}), \text{Nand}(Q, z, R) \rangle$$

It is obvious that G is a normal Δ-approximable model. In order to verify each and every rows of the truth table (as shown in Table 2) it is necessary to define some FzCTL propositions, and then verify them using model checking techniques. According to the first row in the truth table we will investigate validity of following properties.

Property 1 *If $P = 0$ and $R = 1$ then after 2Δ time units, the output is in a constant state like $Q = 1$ and $\bar{Q} = 0$.*

$$(P = 0 \sqcap R = 1) \rightarrow \text{AX}\Big(\text{AX}\big(\text{AG}(Q = 1 \sqcap \bar{Q} = 0)\big)\Big)$$

This proposition (thus property) evaluates to "0" even when increasing time units, by adding more A X *operator to the proposition, its evaluation will not change.*

Property 2 *If $P = 0$ and $R = 1$, once the clock pulse rises, the output is in a constant state like $Q = 1$ and $\bar{Q} = 0$.*

$$(P = 0 \sqcap R = 1) \rightarrow \text{AG}\big(u = 1 \rightarrow (Q = 1 \sqcap \bar{Q} = 0)\big)$$

This proposition also evaluates to "0".

Following trace nullifies Properties 1 and 2.

$$\rightarrow \langle 0, 0, 0, 1, \vartheta, 0, x, y, z, w, Q, \bar{Q} \rangle \rightarrow \langle \Delta, 0, 0, 1, \vartheta, 0, 1, 1, 1, \text{Nand}(z, \vartheta), 1, \text{Nand}(z, Q) \rangle \rightarrow$$
$$\langle 2\Delta, 0, 0, 1, \vartheta, 0, 1, 1, 1, \neg\vartheta, 1, 0 \rangle \rightarrow \cdots \rightarrow \langle A\Delta, 0, 0, 1, \vartheta, 0, 1, 1, 1, \neg\vartheta, 1, 0 \rangle \rightarrow$$
$$\langle 0, 1, 0, 1, \vartheta, 1, 1, 1, 1, \neg\vartheta, 1, 0 \rangle \rightarrow \langle \Delta, 1, 0, 1, \vartheta, 1, 1, 0, \vartheta, \neg\vartheta, 1, 0 \rangle \rightarrow$$
$$\langle 2\Delta, 1, 0, 1, \vartheta, 1, 1, 0, 1, \neg\vartheta, 1, \neg\vartheta \rangle \rightarrow \langle 3\Delta, 1, 0, 1, \vartheta, 1, 1, 0, 1, \ddot{\imath} f \tilde{} \vartheta, 1, 0 \rangle \rightarrow \cdots$$

As can be observed upon raising the clock pulse the value of $\bar{Q}$ resets to "0", yet after 2Δ time units its value changes to $\neg\vartheta$ then in the next step it turns to "0" and stays constant, therefore a hazard occurs on 2Δ time units immediately by the following raising clock pulse when initial value for ϑ is less than "1". Meanwhile by ignoring $\bar{Q}$ in the proposed multi-valued flip-flop the circuit functions correctly and the validity of the following properties is preserved.

Property 3 *If $P = 0$ and $R = 1$ then after 2Δ time units, the output is in a constant state like $Q = 1$.*

$$(P = 0 \sqcap R = 1) \rightarrow \text{AX}\Big(\text{AX}\big(\text{AG}(Q = 1)\big)\Big)$$

Property 4 *If $P = 0$ and $R = 1$, once the clock pulse rises, the output is in a constant state like $Q = 1$.*

$$(P = 0 \sqcap R = 1) \rightarrow \text{AG}\big(u = 1 \rightarrow (Q = 1)\big)$$

Properties 5 and 6 are valid according to the second row in the truth table.

Property 5 *If $P = 1$ and $R = 0$ then after 2Δ time units, the output is in a constant state like $Q = 0$ and $\bar{Q} = 1$.*

$$(P = 1 \sqcap R = 0) \rightarrow \text{AX}\Big(\text{AX}\big(\text{AG}(Q = 0 \sqcap \bar{Q} = 1)\big)\Big)$$

Property 6 *If $P = 1$ and $R = 0$, once the clock pulse rises, the output is in a constant state like $Q = 0$ and $\bar{Q} = 1$.*

$$(P = 1 \sqcap R = 0) \rightarrow \text{A G}\big(u = 1 \rightarrow (Q = 0 \sqcap \bar{Q} = 1)\big)$$

According to the third row in the truth table Property 7 is valid for 3Δ but not less than that.

Property 7 *If $P = 1$ and $R = 1$ then on 3Δ time units immediately following raising the clock pulse, the output is in a constant state like $Q = \vartheta$ and $\bar{Q} = \neg\vartheta$.*

$$(P = 1 \sqcap R = 1) \rightarrow \mathrm{AG}(u - 1 \rightarrow \mathrm{AX}(\mathrm{AX}(\mathrm{AX}(\mathrm{A\,G}(Q = \vartheta \sqcap \bar{Q} = \neg\vartheta)))))$$

Property 8

$$(P = 1 \sqcap R = 1) \rightarrow \mathrm{AG}(u - 1 \rightarrow \mathrm{AX}(\mathrm{AX}(\mathrm{A\,G}(Q = \vartheta \sqcap \bar{Q} = \neg\vartheta))))$$

Property 9

$$(P = 1 \sqcap R = 1) \rightarrow \mathrm{AG}(u - 1 \rightarrow \mathrm{AX}(\mathrm{AG}(Q = \vartheta \sqcap \bar{Q} = \neg\vartheta)))$$

The following trace nullifies Properties 8 and 9:

$$\rightarrow \langle 0, 0, 1, 1, \vartheta, 0, x, y, z, w, Q, \bar{Q}\rangle \rightarrow \langle \Delta, 0, 1, 1, \vartheta, 0, \mathrm{Nand}(y, w), 1, 1, \mathrm{Nand}(z, \vartheta), \mathrm{Nand}(y, \bar{Q}),$$
$$\mathrm{Nand}(z, Q)\rangle \rightarrow \langle 2\Delta, 0, 1, 1, \vartheta, 0, \neg w, 1, 1, \neg\vartheta, \neg\bar{Q}, \neg Q\rangle \rightarrow \langle 3\Delta, 0, 1, 1, \vartheta, 0, d, 1, 1, \neg\vartheta, Q, \bar{Q}\rangle \rightarrow \cdots$$
$$\rightarrow \langle A\Delta, 0, 1, 1, \vartheta, 0, \vartheta, 1, 1, \neg\vartheta, \neg\bar{Q}, \neg Q\rangle \rightarrow \langle 0, 1, 1, 1, \vartheta, 1, \vartheta, 1, 1, \neg\vartheta, \neg\bar{Q}, \neg Q\rangle \rightarrow$$
$$\langle \Delta, 1, 1, 1, \vartheta, 1, \vartheta, \neg\vartheta, \vartheta, \neg\vartheta, Q, \bar{Q}\rangle \rightarrow \langle 2\Delta, 1, 1, 1, \vartheta, 1, \vartheta, \neg\vartheta, \vartheta, \neg\vartheta, \mathrm{Nand}(\neg\vartheta, \bar{Q}), \mathrm{Nand}(\vartheta, Q)\rangle \rightarrow$$
$$\langle 3\Delta, 1, 1, 1, \vartheta, 1, \vartheta, \neg\vartheta, \vartheta, \neg\vartheta, \vartheta, \neg\vartheta\rangle \rightarrow \langle 4\Delta, 1, 1, 1, \vartheta, 1, \vartheta, \neg\vartheta, \vartheta, \neg\vartheta, \vartheta, \neg\vartheta\rangle \rightarrow \cdots$$

Assume rather than the complementary initial values for Q and $\bar{Q}$ they are initialized to values like "0" and "0.5" and the initial value for ϑ equals to "0.25", once the clock pulse rises, Q will gain the following values:

$$0.5 \rightarrow 0 \rightarrow 0.5 \rightarrow 0.25 \rightarrow 0.25 \rightarrow \cdots$$

It can be seen the output become stable in the moment of 3Δ but before that the hazard happens. This problem can be avoided if the initial values for Q and $\bar{Q}$ are complementary values. For complementary values for both Q and $\bar{Q}$ it is necessary to initially preset or clear the circuit. The verification of following property is the proof of not having a hazard.

Property 10

$$(P = 1 \sqcap R = 1 Q = e \sqcap \bar{Q} = \neg e) \rightarrow \mathrm{A}\left((u = 0 \sqcup Q = e \sqcap B = \neg e)\,\mathrm{U}(Q = \vartheta \sqcap B = \neg\vartheta)\right)$$

The above proposition should be checked for each value of e. By adding an attribute called e to the fuzzy program model G and having its value constant in all states it is not necessary to repeat the model checking process for different values of e.

In model checking process while verifying all properties from Properties 1–10, we used a variety of different values from 2^{-6} to 2^{-2} for Δ; modifying Δ did not affect the valuation of propositions. Meanwhile by minimizing Δ both memory usage and turn-around time for model checking process increases. Using OBDD-Vectors (implemented via BuDDy library [LN96]) the corresponding FzKripke for above FzPG is implemented. A verification method for Properties 1–10 is also implemented but the implementation details require a lot of preparation that do not fit in this case study. In this case study, we assumed propagation delays for all gates are equal but it is possible for them to vary, in these situations the valuation for some of propositions such as Property 5 may not equal to "1"; hence it is necessary to define a concrete time generalization for FzPG and FzCTL*.

10. Conclusion and future work

In this paper, not only we introduced Fuzzy Kripke as an extended multi-valued Kripke model but we also defined a temporal fuzzy logic (called FzCTL*) based on FzKripke in order to express temporal properties. FzCTL* include some operators that are rarely seen in other logics. Some modifications are carried out in the meaning of its formulas to avoid shortages in other logics like ambiguity.

The FzKripke complies with classic solutions for the problem of state space explosion. In order to reduce state space, we present two methods; first one is abstraction using bisimulation without affecting the value of logic propositions, and the second one is an approximation technique for both model and logic propositions.

By applying the latter method, state space reduction comes with a controlled error rate for propositions values. A combination of these two methods is also applicable. In another issue, a compact model called FzPG was defined. FzPG is easily convertible to an infinite fuzzy Kripke model, and the infinite FzKripke is also convertible to an approximated finite one using approximation and abstraction techniques together. The FzPG can also be approximated under an especial circumstance.

Although we studied the SFzPG in this paper, it can be proved if the FzPG is not simple then a finite model can be obtained using abstraction and approximation (with an acceptable precision such as Δ), the proof technique is different from what we discussed in this paper. In order to check the temporal properties of a certain FzPG, an approximated Kripke model can be obtained using presented methods; eventually this new model can be processed using conventional symbolic methods. However, it is preferred to directly represent FzPG in a symbolic form without making a Kripke model, and directly calculate its proposition values. For this purpose the same decision diagrams can be used again.

In order to use these models and logics in real applications, some generalizations of them have to be carried out (i.e., FzTA and FzTCTL). In these generalizations the real time must be added in form of fuzzy sets, and the temporal logics must include linguistic expressions alongside formal expressions. Eventually it seems that the fuzzy temporal logic and FzPG proposed in this paper are capable to perform the function checking of even more complex fuzzy circuits. But the corresponding program graph will be more complex and enormous in size; therefore efficient methods will be required to check the properties of these types of system.

References

[AD94] Alur R, Dill DL (1994) A theory of timed automata. Theor Comput Sci 126:183–235

[BCL+94] Burch JR, Clarke EM, Long DE, McMillan KL, Dill DL (1994) Symbolic model checking for sequential circuit verification. IEEE Trans Comput -Aided Des Integr Circuits Syst 13(4):401–424

[BG04] Bruns G, Godefroid P (2004) Model checking with multi-valued logics. In: Lecture notes in computer science, vol 3142. pp 281–293

[BK] Baier C, Katoen J-P (2008) Principles of model checking. MIT Press, Cambridge, p 975. doi:10.1093/comjnl/bxp025

[BPS10] Baresi L, Pasquale L, Spoletini P (2010) Fuzzy goals for requirements-driven adaptation. In: 2010 18th IEEE international requirements engineering conference. IEEE, New York, pp 125–134

[CGD+06] Chechik M, Gurfinkel A, Devereux B, Lai A, Easterbrook S (2006) Data structures for symbolic multi-valued model-checking. Form Methods Syst Des 29(3):295–344

[CGP99] Clarke EM Jr, Grumberg O, Peled DA (1999) Model checking. MIT Press, Cambridge

[Cun04] Cunningham PA (2004) Verification of asynchronous circuits. Ph.D. Thesis, University of Cambridge, Cambridge

[ECD+03] Easterbrook S, Chechik M, Devereux B, Gurfinkel A, Lai A, Petrovykh V, Tafliovich A, Thompson-Walsh C (2003) χChek: a model checker for multi-valued reasoning. In: ICSE '03: Proceedings of the 25th International Conference on Software Engineering. pp 804–805

[EFR74] Epstein G, Frieder G, Rine DC (1974) The development of multiple-valued logic as related to computer science. Computer 7(9):20–32

[Fai05] Fainekos GE (2005) An introduction to multi-valued model checking. Technical Reports (CIS)

[FPS12] Frigeri A, Pasquale L, Spoletini P (2012) Fuzzy time in ltl. arXiv:1203.6278 (arXiv preprint)

[FSGC05] Fern A, Solla AG, Garc J, Cabrer MR (2005) Multi-valued model checking in dense-time. In: Lecture notes in computer science, vol 3751. pp 638–649

[Gur03] Gurfinkel A (2003) Multi-valued symbolic model-checking: fairness, counter-examples, running time. University of Toronto

[Gur10] Gurumurthy KS (2010) Quaternary sequential circuits. IJCSNS 10(7):110–117

[HO89] Hirota K, Ozawa K (1989) The concept of fuzzy flip-flop. IEEE Trans Syst Man Cybern 19(5):980–997

[HP95] Hirota K, Pedrycz W (1995) Design of fuzzy systems with fuzzy flip-flops. IEEE Trans Syst Man Cybern 25(1). doi:10.1109/21.362956

[HR04] Huth M, Ryan M (2004) Logic in computer science: modeling and reasoning about systems, Cambridge University Press, New York. http://www.cs.bham.ac.uk/research/projects/lics/

[Hur84] Hurst SL (1984) Multiple-valued logic? Its status and its future. IEEE Trans Comput C-33(12):1160–1179

[IMMT05] Intrigila B, Magazzeni D, Melatti I, Tronci E (2005) A model checking technique for the verification of fuzzy control systems. In: Proceedings of the international conference on computational intelligence for modelling, control and automation and international conference on intelligent agents, web technologies and internet commerce vol-1 (CIMCA-IAWTIC'06), vol 01. CIMCA '05, Washington, DC, USA. IEEE Computer Society, Silver Spring, pp 536–542

[KHH87] Kameyama M, Hanyu T, Higuchi T (1987) Design and implementation of quaternary NMOS integrated circuits for pipelined image processing. IEEE J Solid-State Circuits 22(1):20–27

[KNSW07] Kwiatkowska M, Norman G, Sproston J, Wang F (2007) Symbolic model checking for probabilistic timed automata. Inf Comput 205(7):1027–1077

[KP03] Konikowka B, Penczek W (2003) On designated values in multi-valued ctl* model checking. Fundam Inf 60(1–4):211–224

[LN96] Lind-Nielsen J (1996) BuDDy—a binary decision diagram package. IT-TR. Department of Information Technology, Technical University of Denmark

[MLL04] Moon S, Lee KH, Lee D (2004) Fuzzy branching temporal logic. IEEE Trans Syst Man Cybern Part B Cybern 34(2):1045–1055

[MP95] Maler O, Pnueli A (1995) Timing analysis of asynchronous circuits using timed automata. In: Camurati PE, Eveking H (eds) Proceedings of CHARME'95, LNCS 987. Springer, Berlin, pp 189–205

[MSSY95] Mateescu A, Salomaa A, Salomaa K, Yu S (1995) Lexical analysis with a simple finite-fuzzy-automaton model. J Univers Comput Sci 1:292–311

[OHK+95] Ozawa K, Hirota K, Koczy LT, Pedrycz W, Ikoma N (1995) Summary of fuzzy flip-flop. In: Proceedings of 1995 IEEE International Conference on Fuzzy Systems, vol 3. doi:10.1109/FUZZY.1995.409897

[OHK96] Ozawa K, Hirota K, Koczy LT (1996) Fuzzy flip-flop. In: Patyra MJ, Mlynek DM (eds) Fuzzy logic. Implementation and applications. Chichester, Wiley, pp 197–236

[Pal00] Palshikar GK (2000) Representing fuzzy temporal knowledge. In: International conference on knowledge-based systems (KBCS-2000), Mumbai, India. pp 252–263

[RA11] Rajaretnam T, Ayyaswamy SK (2011) Fuzzy monoids in a fuzzy finite state automaton with unique membership transition on an input symbol. Int J Math Sci Comput I(I):48–51

[Sla05] Sladoje N (2005) On analysis of discrete spatial fuzzy sets in 2 and 3 dimensions. Ph.D. thesis, Swedish University of Agricultural Sciences Uppsala

[Smi81] Smith KC (1981) The prospects for multivalued logic: a technology and applications view. IEEE Trans Comput C-30(9):619–634

[Smi88] Smith KC (1988) A multiple valued logic: a tutorial and appreciation. Computer, 21(4):17–27

[Tar55] Tarski A (1955) A lattice-theoretical fixpoint theorem and its applications. Pac J Math 5:285–309

[Vaz01] Vazirani VV (2001) Approximation algorithms. Springer, Berlin, p 68

[Wan06] Wang F (2006) Symbolic model checking for probabilistic real-time systems. Ph.D. thesis, University of Birmingham, Birmingham

[Wie10] Wierman MJ (2010) An introduction to the mathematics of uncertainty. Creighton University, p 133. http://aliana.dc.fi.udc.es/files2/MOU.pdf

[WSB+09] Whittle J, Sawyer P, Bencomo N, Cheng BHC, Bruel J-M (2009) Relax: incorporating uncertainty into the specification of self-adaptive systems. In: 2009 17th IEEE international requirements engineering conference, Atalanta, Georgia, August. IEEE, New York, pp 79–88

[WwWG07] Wang L, wen Wang B, Guo Y (2007) Cell mapping description for digital control system with quantization effect. Technical report

Received 6 November 2012

Revised 29 May 2014

Accepted 29 August 2014 by Jin Song Dong

Published online 6 November 2014