Constraint Solving for Verifying Modal Specifications of Workflow Nets with Data

Hadrien Bride[1,2], Olga Kouchnarenko[1,2], and Fabien Peureux[1]

[1] Institut FEMTO-ST – UMR CNRS 6174, Univ. Bourgogne Franche-Comté
16, route de Gray, 25030 Besançon, France
{hbride,okouchna,fpeureux}@femto-st.fr
[2] Inria Nancy Grand Est – CASSIS Project
Campus Scientifique, BP 239, 54506 Vandœuvre-lès-Nancy cedex
{hadrien.bride,olga.kouchnarenko}@inria.fr

Abstract. For improving efficiency and productivity companies are used to work with workflows that allow them to manage the tasks and steps of business processes. Furthermore, modalities have been designed to allow loose specifications by indicating whether activities are necessary or admissible. This paper aims at verifying modal specifications of coloured workflows with data assigned to the tokens and modified by transitions. To this end, executions of coloured workflow nets are modelled using constraint systems, and constraint solving is used to verify modal specifications specifying necessary or admissible behaviours. An implementation supporting the proposed approach and promising experimental results on an issue tracking system constitute a practical contribution.

Keywords: Workflows, Modalities, Coloured Petri Nets, Constraint System.

1 Introduction

To improve efficiency and productivity companies are used to work with workflows describing the set of possible runs of a particular system/process. The development of such workflows has become a crucial part of companies effort since they define the organisational core of these companies by increasing their business agility, flexibility and efficiency. Major Key Performance Indicators (compliance with respect to regulations and directives, end-user acceptance and confidence, etc.) are often directly determined by the quality of the workflows in use, and therefore much of the companies successes depends on them. From this, it requires workflow specifications to be properly designed and carefully verified to ensure they comply with the expected and needed workflows properties. However, the increasing complexity of such workflows makes them error-prone and the verification of the related models still remains a tough task [1].

Many modelling languages and related tooling to describe workflow systems have been proposed [2]. Among them, workflow Petri nets (WF-nets for short) [3] are well suited for modelling and analysing discrete event systems exhibiting behaviours such as concurrency, conflict, and causal dependency between events.

They represent finite or infinite-state processes, and several important verification problems, such as reachability or soundness, are known to be decidable. However, due to the growing complexity of modeled processes, WF-nets describing them tend to be too complex and extremely large [4]. Moreover, WF-nets do not model the data often relevant to address realistic processes [5]. To handle data, workflows can be modelled by coloured Petri nets where data are assigned to the tokens and can be modified by transitions based on their contents [6].

Within refinement approaches for workflow development, modal specifications [7] have been designed to allow *loose* specifications by imposing restrictions on the possible refinements by indicating whether activities–transitions in the case of WF-nets–are *necessary* or *admissible*. Modalities provide a flexible tool for workflow development as decisions can be delayed to later steps of the development life cycle, when performing workflow refinements.

The paper first presents modal specifications with additional constraints on the initial state of the workflow as well as with conditions on coloured transitions and their causalities, i.e. on activities. Second, it defines a formal framework based on constraint systems to model executions of CWF-nets, which, in turn, enables the automated verification of modal specifications. Third, it reports on an implementation of the approach, which is successfully experimented on a concrete case study to validate an issue tracking system.

After providing preliminaries on Petri nets, coloured Petri nets and modal specifications in Sect. 2, Sect. 3 introduces the *Questions and Answers Portal* motivating example and specifies it as a CWF-net. The main contribution in Sect. 4 consists of a formal framework based on constraint systems to model executions of CWF-nets and their structural properties, as well as to verify their modal specifications. An implementation supporting the proposed approach and promising experimental results constitute a practical contribution in Sect. 5. Finally, Sect. 6 concludes the paper by discussing related work and future work.

2 Background

This section presents preliminaries on Petri nets, coloured Petri nets [8] and introduces modal specifications based on proposals in [9] and [10].

2.1 Petri Nets

Petri nets are a basic model of parallel and distributed systems defined as follow.

Definition 1 (Petri net). *A Petri net is a tuple (P, T, F) where P is a finite set of places, T is a finite set of transitions ($P \cap T = \emptyset$), and $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs.*

A Petri net with arcs of weight 1 (i.e. every element of F is unique) is called an *ordinary* Petri net. Let $g \in P \cup T$ and $G \subseteq P \cup T$. We use the notations: $g^\bullet = \{g' | (g, g') \in F\}$, ${}^\bullet g = \{g' | (g', g) \in F\}$, $G^\bullet = \cup_{g \in G}\, g^\bullet$, and ${}^\bullet G = \cup_{g \in G}\, {}^\bullet g$. These definitions allow characterizing structural features such as siphons and traps.

Definition 2 (Siphon/Trap). *Let $N \subseteq P$ such that $N \neq \emptyset$: N is a trap if and only if $N^\bullet \subseteq {}^\bullet N$, and N is a siphon if and only if ${}^\bullet N \subseteq N^\bullet$.*

Lemma 1 ([11]). *A marked trap cannot be unmarked, and an unmarked siphon cannot be marked.*

Theorem 1 ([11]). *An ordinary Petri net without siphon is live.*

Coloured Petri nets [8] are high-level Petri nets where data assigned to the tokens can be modified by transitions based on their contents. Let Ξ be a non-empty set of *data-types* (called colours), where each *data-type* is a set of *data-values*. We denote here $\mathcal{L}(\mathcal{V}, \mathcal{W})$ the space of linear maps from $\mathcal{V}$ to $\mathcal{W}$, and $\mathcal{O}$ the zero map.

Definition 3 (Coloured Petri net). *A coloured Petri net (CPN) is a tuple (P, T, C, W) where:*
- *P is a finite set of places, T is a finite set of transitions, such that $P \cap T = \emptyset$,*
- *$C : P \cup T \to \Xi$ is the colour-function,*
- *$W^- : P \times T \to \mathcal{L}(\Xi, \Xi)$ is the pre-incidence function,*
- *$W^+ : P \times T \to \mathcal{L}(\Xi, \Xi)$ is the post-incidence function.*

A *marking* of a CPN is a function M defined on P, such that $\forall p \in P, M(p) \in C(p) \to \mathbb{N}$. Two markings M_a and M_b are in relation $M_a \geq M_b$ if and only if $\forall p \in P, \forall c \in C(p), M_a(p)(c) \geq M_b(p)(c)$.

A weighted set of transitions is a function x defined on T, such that $\forall t \in T, x(t) \in C(t) \to \mathbb{N}$. From now on, let $\circledast$ denote the generalized matrix-multiplication where each product is replaced by a function composition. With this notation, a transition defined by $x(t) \in C(t) \to \mathbb{N}$ is *enabled* in a marking M_a if and only if $M_a \geq W^-(t) \circledast x(t)$. When $x(t)$ is *enabled*, it may *fire*. If $x(t)$ *fires*, a new marking $M_b = M_a + (W^+ - W^-)(t) \circledast x(t)$ is reached. M_b is said to be *directly reachable* from M_a by transition $x(t)$, written $M_a \xrightarrow{x(t)} M_b$. Let *reachability* relation be the reflexive and transitive closure of the *direct reachability*.

Let $\sigma = x_1(t_1), .., x_n(t_n)$ be a sequence of transitions, i.e. $\forall i \in 1..n, x_i(t_i) \in C(t_i) \to \mathbb{N}$, we say that σ is a valid sequence of transitions with respect to the weighted set of transitions x, denoted $\sigma \models x$, if $\forall t \in T, x(t) = \sum_{i|t_i=t} x_i(t_i)$.

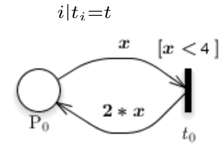


Fig. 1. Example of a CPN

For example, let $\Xi = \{C_1, C_2\}$ where $C_1 = \{1, 2, 3, 4, 5, 6\}$ and $C_2 = \{1, 2, 3\}$. For the CPN_1 of Fig. 1, let $C(P_0) = C_1$ and $C(t_0) = C_2$. Let be $x \in C_2$ and $t_0(x)$ a transition such that $\forall e \in C_2 \setminus \{x\}, t_0(x)(e) = 0$ and $t_0(x)(x) = 1$. When transition $t_0(x)$ fires, it consumes a token x and produces a token $x * 2$ in P_0. Let $M_{CPN_1}(x)$ be the marking such that $\forall e \in C_1 \setminus \{x\}, M_{CPN_1}(x)(P_0)(e) = 0$ and $M_{CPN_1}(x)(P_0)(x) = 1$. For $\sigma = t_0(1), t_0(2)$, we have $M_{CPN_1}(1) \xrightarrow{t_0(1)} M_{CPN_1}(2) \xrightarrow{t_0(2)} M_{CPN_1}(4)$. Let wt be the weighted set of transitions such that $wt(t_0)(1) = wt(t_0)(2) = 1$ and $wt(t_0)(3) = 0$, we have $\sigma \models wt$.

2.2 Coloured Workflow Nets

From the framework of coloured Petri nets, we now define coloured workflow nets (CWF-nets for short).

Definition 4 (CWF-net). *A coloured Petri net (P, T, C, W) is a CWF-net if and only if: PN have two special places i and o where ${}^{\bullet}i = \emptyset$ and $o^{\bullet} = \emptyset$, and for each node $n \in (P \cup T)$ there exists a path from i to o passing through n.*

In the rest of the paper, the following notations are used:

- M_i: the set of initial markings of a CWF-net where $\forall M_a \in M_i, M_a(i) \neq \mathcal{O}$ and $\forall p \in P \setminus i, M_a(p) = \mathcal{O}$,
- M_o: the set of final markings of a CWF-net where $\forall M_a \in M_o, M_a(o) \neq \mathcal{O}$ and $\forall p \in P \setminus o, M_a(p) = \mathcal{O}$,
- $M_1 \xrightarrow{\sigma} M_n$: for $\sigma = x_1, x_2, ..., x_{n-1}$, there are markings such that $M_1 \xrightarrow{x_1} M_2 \xrightarrow{x_2} ... \xrightarrow{x_{n-1}} M_n$,
- $M_a \xrightarrow{*} M_b$: there exists σ such that $M_a \xrightarrow{\sigma} M_b$.

In our approach, constraints over markings and weighted sets of transitions are expressed using Presburger arithmetic [12] in order to remain within the realm of decidability. Let $M_a(p)$ be the marking of a place p, and f_p a first-order formula over Presburger arithmetic with free variables over $C(p)$. We denote by $M_a(p) \models f_p$ the fact that $M_a(p)$ satisfies f_p, i.e. $(\bigwedge_{d \in C(p)} d = M_a(p)(d)) \wedge f_p$ is satisfiable. Similarly, let $x_a(t)$ be the weighted set of transitions t, and f_t a first-order formula over Presburger arithmetic formulae with free variables over $C(t)$. We write $x_a(t) \models f_t$ when $x_a(t)$ satisfies f_t, i.e. $(\bigwedge_{d \in C(t)} d = x_a(t)(d)) \wedge f_t$ is satisfiable.

To illustrate this notation on CPN_1 of Fig. 1, let $f_{P_0} = (C_1(1) = 1)$ be a formula over Presburger arithmetic with free variables over $C(P_0)$. We have $M_{CPN_1}(1) \models f_{P_0}$, which expresses the fact that the marking $M_{CPN_1}(1)$ contains exactly 1 token of value 1. Likewise, let $f_{t_0} = (C_2(2) = 1)$ be a formula with free variables over $C(t_0)$, and wt the weighted set of transitions such that $t_0(2) \models wt$. We have $wt \models f_{t_0}$ expressing the fact that wt is valid with respect to a sequence of transitions containing a transition $x(t_0)$ where $x(t_0)(2) = 1$.

An execution between markings M_a and M_b of a CWF-net is a sequence of transitions σ such that $M_a \xrightarrow{\sigma} M_b$. An execution is a *correct* execution if and only if $M_a \in M_i$ and $M_b \in M_o$. The behaviour of a CWF-net is defined as the set Σ of all its *correct* executions.

2.3 CWF-nets with Modalities

Modal specifications permit specifiers to indicate that a transition is *necessary* or just *admissible*. In the context of CWF-nets, it usually means that there are two kinds of transitions: the *must*-transitions and the *may*-transitions. A *may*-transition (resp. *must*-transition) is a transition fired by at least one *correct* execution (resp. by all the *correct* executions) of a CWF-net.

We extend this concept to allow specifiers to indicate modal properties on several transitions and on their causalities. We also add the possibility to parameterize transitions as well as the initial marking, to permit a precise modal specification of desired behavior.

Definition 5 (Well-formed modal formula). *Let $CPN = (P, T, C, W)$ be a CWF-net. The language S of well-formed* modal *specification formulae is defined by the following grammar of axiom A, where $t \in T$ and p (resp. q) is a first-order formula over Presburger arithmetic formulae [12] with free variables over $C(t)$ (resp. $C(i)$):* $\quad A \to [q]B \quad , \quad B \to (B \wedge B)|(B \vee B)|(\neg B)|t[p] \quad .$

These formulae allow specifiers to express modal properties about CWF-nets' correct executions. Any modal specification formula $[q]m \in S$ can be interpreted as a *may*-formula or a *must*-formula. Given a CWF-net, a *may*-formula (resp. a *must*-formula) describes a behaviour constrained by m that has to be ensured by at least one (resp. all) correct execution of initial state satisfying q. Formally, the semantics of a formula m generated from B, where the semantics of $\neg, \vee$ and $\wedge$ is standard, is defined by:

- $wt \models_{may} t[p]$ iff $\exists \sigma = x_1, x_2, ..., x_{n-1} \in \Sigma.\ \sigma \models wt \wedge \exists k.\ x_k(t) \models p$,
- $wt \models_{must} t[p]$ iff $\forall \sigma = x_1, x_2, ..., x_{n-1} \in \Sigma.\ \sigma \models wt \wedge \exists k.\ x_k(t) \models p$.

Furthermore, given a *may*-formula (resp. *must*-formula) $[q]m \in S$, its semantics is inductively defined by:

- $CPN \models_{may} q[m]$ iff $\exists \sigma \in \Sigma, M_a \in M_i, M_b \in M_o.\ M_a \xrightarrow{\sigma} M_b \wedge M_a(i) \models q \wedge \sigma \models wt \wedge wt \models_{may} m$,
- $CPN \models_{must} q[m]$ iff $\forall \sigma \in \Sigma, M_a \in M_i . M_a(i) \models q . (\exists M_b \in M_o.\ M_a \xrightarrow{\sigma} M_b \Rightarrow (\sigma \models wt \wedge wt \models_{must} m))$.

Definition 6 (Modal Specification). *A modal specification is defined by a tuple (M_{may}, M_{must}) where $M_{may} \subset S$ is a finite set of may-formulae, and $M_{must} \subset S$ is a finite set of must-formulae.*

A CWF-net CPN satisfies a modal specification $MS = (M_{may}, M_{must})$, written $CPN \models MS$, iff $\forall m \in M_{may}.\ CPN \models_{may} m \wedge \forall m' \in M_{must}.\ CPN \models_{must} m'$.

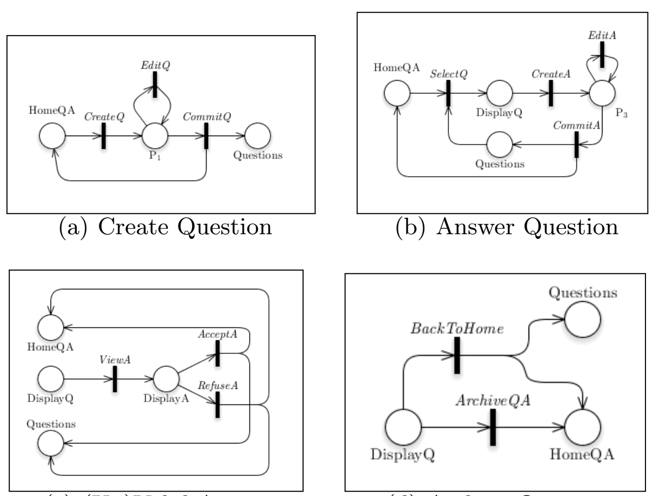
2.4 Constraint System

A constraint system is defined by a set of constraints (properties), which must be satisfied by the solution of the problem it models. Such a system can be represented as a Constraint Satisfaction Problem (CSP) [13]. Formally, a CSP is a tuple $\Omega =< X, D, C >$ where X is a set of variables $\{x_1, \ldots, x_n\}$, D is a set of domains $\{d_1, \ldots, d_n\}$, where d_i is the domain associated with the variable x_i, and C is a set of constraints $\{c_1(X_1), \ldots, c_m(X_m)\}$, where a constraint c_j involves a subset X_j of the variables of X. It is such that each variable appearing in a constraint should take its value from its domain. Hence, a CSP models NP-complete problems as search problems where the corresponding search space is the Cartesian product space $d_1 \times \ldots \times d_n$.

The solution of a CSP Ω is computed by a labelling function $\mathcal{L}$, which provides a set v (called valuation function) of tuples assigning each variable x_i of X to one value from its domain d_i such that all the constraints of C are satisfied. More formally, v is consistent—or satisfies a constraint $c(X)$ of C—if the projection of v on X is in $c(X)$. If v satisfies all the constraints of C, then Ω is a consistent or satisfiable CSP. In the present paper, we propose to use Constraint Logic Programming over Finite Domains, written CLP(FD) [14], to solve the CSP representing the modal specifications to be verified.

3 Motivating Example

Let us consider a business process workflow of a *Question and Answer* portal, which is a part of a proprietary issue tracking system used to manage bugs and issues requested by the customers of a tool provider company[3]. It allows company's customers to ask questions that are then answered by the company's sellers. To use the system, users have to be registered. Three types of users can log-in: clients, sellers and administrators. Clients can ask questions that are then answered by sellers. Once the answer to a question has been validated by the client who asked it, the administrator archives the question. An execution of the workflow is complete once all users have been logged-out and unregistered. We present one of the several refinements of the workflow modelled by a CWF-net. For clarity, this CWF-net is described by several *sub*-CWF-nets where the places with the same name are the same, as, e.g., $HomeQA$ place in Fig. 3 and 2(a).



(a) Create Question (b) Answer Question

(c) (Un)Valid Answer (d) Archive Question

Fig. 2. *sub*-CWF-nets of the *Question and Answer* CWF-net

In this refinement, there are three colours. The first colour models a set $U = \{u_1, .., u_t\}$ of t user names representing the different users of the system.

[3] For confidentiality reasons, the details about this case study are not given.

The second colour is used for a set $R = \{client, seller, admin\}$ of roles, which are assigned to users. Finally, to represent the question states, the third colour $Q = \{unanswered, answered, validated\}$ is a set of statuses. Table 1 shows the colours associated with places of the *Question and Answer* CWF-net, and Tab. 2 shows the colours, inputs, outputs and guards ($u, u_1, u_2 \in U, r \in R$, $q \in Q$).

An execution of the *Question and Answer* CWF-net starts with at least three users (a client, a seller, and an administrator). To illustrate how this CWF-net works, let us consider the following execution with u_1, u_2 and u_3 as initial marking: each user is registered, then logs in and navigates to the QA's Home(Fig. 3):

- $Register(u_1, client)$, $Login(u_1, client)$, $HomeToQA(u_1, client)$
- $Register(u_2, seller)$, $Login(u_2, seller)$, $HomeToQA(u_2, seller)$
- $Register(u_3, admin)$, $Login(u_3, admin)$, $HomeToQA(u_3, admin)$

The client creates a new question (Fig. 2(a)):

- $CreateQ(u_1, client)$, $CommitQ(u_1, client, unanswered, u_1)$

The seller selects the question and the answer (Fig. 2(b)):

- $SelectQ(u_2, seller, unanswered, u_1)$, $CreateA(u_2, seller, unanswered, u_1)$
- $CommitA(u_2, seller, answered, u_1)$

The client selects the question, reads the answer and validates (Fig. 2(c)):

- $SelectQ(u_1, client, answered, u_1)$, $ViewA(u_1, client, answered, u_1)$
- $AcceptA(u_1, client, answered, u_1)$

The administrator selects the question and archives it (Fig. 2(d)):

- $SelectQ(u_3, admin, validated, u_1)$,$ArchiveQA(u_3, admin, validated, u_1)$

The users navigate to Home and then log-out and are unregistered (Fig. 3):

- $QAtoHome(u_1, client)$, $Logout(u_1, client)$ $UnRegister(u_1, client)$
- $QAtoHome(u_3, seller)$, $Logout(u_3, seller)$ $UnRegister(u_2, seller)$
- $QAtoHome(u_3, admin)$, $Logout(u_3, admin)$ $UnRegister(u_3, admin)$

Regarding this business process, the goal is to verify, at the specification or design stage of the development, some required behavioural properties derived from textual requirements and business analyst expertise. We consider the following properties, denoted p_i for later references ($nbUsers$ denotes the number of users in the initial marking: $nbUsers = \sum_{r=1}^{t} M_i(i)(r)$).

p_1: $QA \models_{must} [true]Register[u = u_1] \wedge .. \wedge Register[u = u_t]$: all users must register;

p_2: $QA \models_{may} [true]ViewA[r = admin]$: an admin may view an answer;

p_3: $QA \models_{may} [true]CreateQ[r = client, u = u_x] \wedge RefuseA[r = client, u_1 = u_x]$: u_x client may create a question and refuse the answer;

p_4: $QA \models_{must} [true]CommitQ[u_2 = u_x] \Rightarrow CommitA[u_2 = u_x] \wedge ArchiveQA[u_2 = u_x]$: when u_x asks a question it must be answered and archived;

p_5: $QA \models_{must} [true]\neg CreateA[r = client]$: a client must not answer a question;

p_6: $QA \models_{may} [nbUsers > 3]CreateQ[u = u_x] \wedge \neg CreateQ[u = u_y]$: there may be an user u_x who asks a question while another (i.e. u_y) does not;

p_7: $QA \models_{must} [nbUsers < 3]\neg CreateQ[true]$: if there is less than three users, no question is asked;

p_8: $QA \models_{must} [true]CreateQ[true] \Rightarrow (Register[r = client] \wedge Register[r = seller] \wedge Register[r = admin])$: if a question is asked then the system must have registered a client, a seller and an administrator.

Let us emphasize that these properties could not be expressed without taking colours into account because data are necessarily involved.

Table 1. Colours of *Question and Answer* CWF-net's Places

Colours	Places
U	i, o
$U \times R$	$P_0, Home, HomeQA, HomeSR$
$U \times R \times Q \times U$	$DisplayQ, DisplayA, P_1, P_3$
$Q \times U$	$Questions$

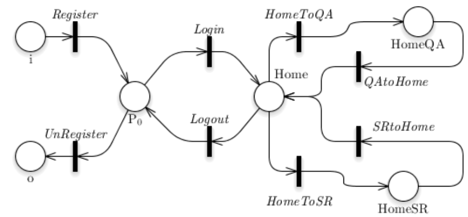


Fig. 3. Login and Navigation *sub*-CWF-net

Table 2. Colours, Inputs, Outputs, and Guards of *Question and Answer* CWF-net's Transitions

Transition	Colours	Inputs	Outputs	Guard
$Register$	$U \times R$	u	(u, r)	$True$
$UnRegister$	$U \times R$	(u, r)	u	$True$
$Login, Logout$	$U \times R$	(u, r)	(u, r)	$True$
$HomeToQA$	$U \times R$	(u, r)	(u, r)	$True$
$QAtoHome$	$U \times R$	(u, r)	(u, r)	$True$
$HomeToSR$	$U \times R$	(u, r)	(u, r)	$True$
$SRtoHome$	$U \times R$	(u, r)	(u, r)	$True$
$CreateQ$	$U \times R$	(u, r)	$(u, r, 'unanswered', u)$	$r =' client'$
$EditQ$	$U \times R \times Q \times U$	(u_1, r, q, u_2)	(u_1, r, q, u_2)	$True$
$CommitQ$	$U \times R \times Q \times U$	(u_1, r, q, u_2)	(u_1, r) and (q, u_2)	$True$
$SelectQ$	$U \times R \times Q \times U$	(u_1, r) and (q, u_2)	(u, r, q, u)	$True$
$CreateA$	$U \times R \times Q \times U$	(u_1, r, q, u_2)	$(u_1, r, 'answered', u_2)$	$r =' seller' \wedge q =' unanswered'$
$EditA$	$U \times R \times Q \times U$	(u_1, r, q, u_2)	(u_1, r, q, u_2)	$True$
$CommitA$	$U \times R \times Q \times U$	(u_1, r, q, u_2)	(u_1, r) and (q, u_2)	$True$
$ViewA$	$U \times R \times Q \times U$	(u_1, r, q, u_2)	(u_1, r, q, u_2)	$u_1 = u_2 \wedge q =' answered'$
$AcceptA$	$U \times R \times Q \times U$	(u_1, r, q, u_2)	(u_1, r) and $('validated', u_2)$	$True$
$RefuseA$	$U \times R \times Q \times U$	(u_1, r, q, u_2)	(u_1, r) and $('unanswered', u_2)$	$True$
$ArchiveQA$	$U \times R \times Q \times U$	(u_1, r, q, u_2)	(u_1, r)	$r =' admin' \wedge q =' validated'$
$BackToHome$	$U \times R \times Q \times U$	(u_1, r, q, u_2)	(u_1, r) and (q, u_2)	$True$

4 Modelling Executions of CWF-nets

This section aims to model the correct executions of a CWF-net by a constraint system, which is then solved to validate or invalidate properties of interest.

Theorem 2 (State equation [8]). *Let $CPN = (P, T, C, W)$, if a marking M_b is reachable from M_a then there exists x a weighted set of transitions such that:*

$$M_b = M_a + (W^+ - W^-) \circledast x. \tag{1}$$

To illustrate (1), let us consider the CWF-net described in Sect. 3. Let M_1 be a marking such that $M_1(i) = \{u_1\}, \forall p \in P \setminus \{i\}.M_1(p) = \mathcal{O}$, and M_2 be a marking such that $M_2(o) = \{u_1\}, \forall p \in P \setminus \{i\}.M_2(p) = \mathcal{O}$. The marking M_2 is reachable from M_1 by the transition sequence $\alpha = \mathit{Register}(u_1, \mathit{client})$, $\mathit{Unregister}(u_1, \mathit{client})$. Let x_1 denote the weighted set of transitions in α, then we have $M_2 = M_1 + (W^+ - W^-) \circledast x_1$.

The set of solutions of the state equation (1) of a CWF-net, where $M_a \in M_i$ and $M_b \in M_o$, defines an over-approximation of the set of its correct executions. A solution of the state equation (1) is called *spurious* if it does not correspond to an execution of the considered CWF-net. For example, let us now consider the weighted set x_2 of the transitions $\mathit{Register}(u_1, \mathit{client})$, $\mathit{Unregister}(u_1, \mathit{client})$, and $\mathit{EditQ}(u_1, \mathit{client}, \mathit{unanswered}, u_1)$. In this case we have $M_2 = M_1 + (W^+ - W^-) \circledast x_2$, however the weighted set of transitions x_2 does not correspond to any correct execution, i.e. x_2 is a spurious solution. This is because of the transition EditQ, which produces and consumes the same token in place P_1.

To dismiss spurious solutions, this over-approximation can be refined by considering structural properties of the places and transitions involved in the considered executions. To this end, we introduce the notion of the subnet of a CWF-net associated with a solution of its state equation (1).

Definition 7. *Let $CPN = (P, T, C, W)$ a CWF-net, M_a, M_b two markings of CPN, and x a weighted set of transitions such that $M_b = M_a + (W^+ - W^-) \circledast x$. We define the subnet $sCPN(x) = (sP, sT, sF)$ where:*

- $sP = \{p \in P \setminus \{p \in P | M_a(p) \neq \mathcal{O} \vee M_b(p) \neq \mathcal{O}\} \mid \exists t \in T, W^+(t, p) \circledast x(t) > 0 \vee W^-(p, t) \circledast x(t) > 0\}$
- $sT = \{t \in T \mid x(t) > 0\}$
- $sF = \{(a, b) \mid a \in (sP \cup sT) \wedge b \in (sP \cup sT) \wedge (W^+(a, b) \circledast x(a) > 0 \vee W^-(a, b) \circledast x(b) > 0)\}$

Among various structural properties of CWF-nets, the existence of a siphon and a trap in the subnet of a CWF-net, associated with a solution of its state equation (1), is relevant (Lemma 1). Moreover, any subnet of a solution of (1) that contains a siphon or a trap is a spurious solution. Theorem (3) defines a constraint system for determining the presence of a siphon in a Petri net.

Theorem 3 ([10]). *Let $\theta(PN)$ be the following constraint system associated with a Petri net $PN = (P, T, F)$: $\forall p \in P, \forall t \in^\bullet p. \sum_{p' \in^\bullet t} \xi(p') \geq \xi(p) \wedge \sum_{p \in P} \xi(p) > 0$ where $\xi : P \to \{0, 1\}$ is a valuation function. PN contains a siphon if and only if there is a valuation satisfying $\theta(PN)$.*

In this way, checking the existence of traps and of siphons can be done simultaneously thanks to the following theorem.

Theorem 4. *Let $CPN = (P, T, C, W)$ a CWF-net, M_a, M_b two markings, and x a weighted set of transitions such that $M_b = M_a + (W^+ - W^-) \circledast x$. If $sCPN(\nu)$ contains a trap (resp. siphon) N then N is also a siphon (resp. trap).*

Structural properties (the siphon existence) can be exploited to refine the state equation (1) over-approximation. Let us consider the above-mentioned spurious solution x_2. The subnet of x_2 is shown in Fig. 4. We can see that in this subnet formed by the solution x_2, place P_1 is a siphon as the valuation ξ, such that $\xi(P_1) = 1$ and $\xi(P_0) = 0$, satisfies $\theta(sCPN(x_2))$.

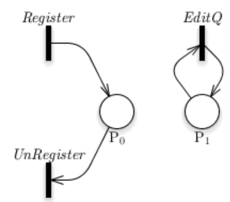


Fig. 4. Subnet of x_2

Theorem 5 uses the state equation (1) together with the constraint system of Theorem 4 to provide a constraint system for modeling executions of CWF-net without spurious solutions.

Theorem 5. *Let $CPN = (P, T, C, W)$ a CWF-net, M_a, M_b two markings of CPN, x a weighted set of transitions, and $sCPN(x) = (sP, sT, sF)$ the subnet associated to CPN and the weighted set of transitions x. Let $\phi(CPN, M_a, M_b, x)$ be the following constraint system:*
- $M_b = M_a + (W^+ - W^-) \circledast x$,
- *there is no valuation satisfying $\theta(sCPN(x))$, and*
- $\forall p \in sP, |^\bullet p| \leq 1 \wedge |p^\bullet| \leq 1$,

If $\phi(CPN, M_a, M_b, x)$ is satisfiable then there exists $\sigma \models x$ such that $M_a \xrightarrow{\sigma} M_b$.

Let $CPN = (P, T, C, W)$ be a CWF-net, M_a, M_b two markings of CPN, the set of solutions of $\phi(CPN, M_a, M_b, x)$ is an under-approximation of the set of correct executions reaching M_b from M_a in CPN. Any execution modelled by the constraint system ϕ is called a segment. Any correct execution of CPN can be modelled by a finite number of segments.

Theorem 6. *Let $CPN = (P, T, C, W)$ a CWF-net, M_a, M_b two markings of CPN. Let $\psi(CPN, M_a, M_b, X, K)$ be the following constraint system:*
- $\forall k \in 1..K, \phi(CPN, M_{k-1}, M_k, x_k)$,
- $M_0 = M_a \wedge M_K = M_b$, *and*
- $X = \{x_1, .., x_K\}$.

There exists $\sigma = \sigma_1, .., \sigma_K$ such that $\forall i \in 1..K, \sigma_i \models x_i$ and $M_a \xrightarrow{\sigma} M_b$ if and only if $\exists K \in \mathbb{N}$ such that $\psi(CPN, M_a, M_b, X, K)$ is satisfiable.

The constraint system of Theorem (6) allows modelling any correct execution of a CWF-net composed of at most K segments. This naturally leads us to consider two decision problems.

The first decision problem, called the K*-bounded validity of a modal formula*, only considers executions formed by K segments, at most. The second one, called the *unbounded validity of a modal formula*, generalizes the first problem by considering executions formed by an arbitrary number of segments.

To verify the K*-bounded* validity of a modal $[q]m$ *may*-formula determining the existence of a correct execution modelled by K segments starting from an initial marking satisfying q where the behaviour of m is satisfied, is enough. Similarly, determining the K*-bounded* validity of a modal $[q]m$ *must*-formula can be done by determining the non-existence of a correct execution modelled by K segments starting from an initial marking satisfying q where the behaviour of $\neg m$ is satisfied.

Let x be a weighted set of transitions, and $[q]m$ a modal formula. We denote $P(x, m)$ the constraint corresponding to the formula m. To construct this constraint, every terminal symbol $t[p]$ of the formula m is replaced by the corresponding constraint obtained by replacing every free variable of p in $C(t)$ by the corresponding variable over x. To illustrate this construction, let us consider $m = CreateQ[r = client, u = u_x] \wedge RefuseA[r = client, u_1 = u_x]$. The corresponding constraint $P(x, m)$ is $x(CreateQ)(r) = client \wedge x(CreateQ)(u) = u_x \wedge x(RefuseA)(r) = client \wedge x(RefuseA)(u) = u_x$. We say that $x \models m$ (i.e. x satisfies m) if and only if $x \wedge P(x, m)$ is satisfiable. Let $X = \{x_1, .., x_n\}$ be a set of weighted sets of transitions, $X \models m$ if and only if $x_1 \models m \vee .. \vee x_n \models m$.

Theorem 7. *Let CPN be a CWF-net and $M = (M_{may}, M_{must})$ be a modal specification of CPN. CPN satisfies the modal specification M if and only if:*
- *$\forall [q]m \in M_{may}$ $\exists k \in \mathbb{N}$, $M_a \in M_i$ and $M_b \in M_o$ such that*
  $M_a(i) \models q \wedge \psi(CPN, M_a, M_b, X, K) \wedge X \models m$ is satisfiable.
- *$\forall [q]m \in M_{must}$ $\exists k \in \mathbb{N}$, $M_a \in M_i$ and $M_b \in M_o$ such that*
  $M_a(i) \models q \wedge \psi(CPN, M_a, M_b, X, K) \wedge X \models \neg m$ is not satisfiable.

Theorem (7), with $k \leq K$, defines a constraint system, which allows to determine the K*-bounded* validity of a modal specification.

Theorem 8. *Let CPN be a CWF-net where Ξ is composed of finite data-types, $\bar{R}_{must}$ the set of all well-formed must-formulae not satisfied by CPN, and R_{may} the set of all well-formed may-formulae satisfied by CPN. There exists K_{max} such that:*
- *$\forall [q]m \in \bar{R}_{must}$ $\exists k \leq K_{max}$, $M_a \in M_i$ and $M_b \in M_o$ such that*
  $M_a(i) \models q \wedge \psi(CPN, M_a, M_b, X, K) \wedge X \models \neg m$ is satisfiable.
- *$\forall [q]m \in R_{may}$ $\exists k \leq K_{max}$, $M_a \in M_i$ and $M_b \in M_o$ such that*
  $M_a(i) \models q \wedge \psi(CPN, M_a, M_b, X, K) \wedge X \models m$ is satisfiable.

Theorem (8) states that for any CWF-net where Ξ is composed of finite *data-types*, there exists K_{max} such that the K_{max}*-bounded* validity of a modal specification is equivalent to the *unbounded* validity of a modal specification. However this is not true for CWF-net where Ξ is composed of infinite *data-types*. This is consistent with the fact that *reachability* of CPN with infinite colours is undecidable as they can, for example, simulate a Minsky 2-counter machine [15].

5 Implementation and Experiments

This section describes the tool chain developed to experimentally validate this paper's proposals, and illustrates its use on the motivating example from Sect. 3.

5.1 Overview of the Prototype Architecture and Procedures

In order to assess our work, especially regarding its feasibility and efficiency, we have implemented our approach within the Eclipse platform on a trial basis. The process starts using a graphical CWF-net editor created within the Sirius framework[4], which is an EMF-based open source project to create customized graphical modeling workbench by leveraging Eclipse Modeling technologies. Basically, it provides a generic workbench for model-based architecture engineering that could be easily tailored to fit the specific needs of a given Domain Specific Language, e.g., CWF-nets in our context. Hence the developed CWF-net editor allows producing an XML file corresponding to the designed CWF-net model. It is completed by the modal specification, which is manually designed using a dedicated XML format. Once syntactically and semantically validated by a modal checker, these inputs are translated into constraint systems that are handled by the CLP(FD) library of Sicstus Prolog[5]. Finally, a report is generated.

To verify a *may*-formula (resp. a *must*-formula) $[q]m$, the tool first checks if there exists a solution x of the over-approximation, given by the state equation (Theorem (2)) for which the subnet (Theorem (7)) does not contain siphons, such that the modelled execution satisfies (resp. does not satisfy) $[q]m$ (we denote this constraint system φ). If such an execution exists, it then tries to find an execution modelled by K segments (Theorem (6)), which satisfies (resp. does not satisfy) $[q]m$ (we denote this constraint system $\phi(K)$). It then reports about the K-bounded validity of a given modal formula m. To cope with the complexity raised by K_{max}, K can be fixed to a manageable value. When fixing K to K_{max} (or greater than K_{max}), the algorithm enables deciding the unbounded validity of the *must*-formula m. The results given in Sect. 4 ensure its soundness and completeness. Finally, solving a CSP over a finite domain being an NP-complete problem with respect to the domain size, this algorithm inherits this complexity.

Modellers often use, in the context of workflow development, infinite colours (e.g., strings, integers) to represent data (e.g., usernames of a system, identifiers of files), even if these data are usually not directly manipulated by the control flow. However, CWF-nets with infinite colours cannot be directly handled due to the nature of the constraint solver over finite domains. Fortunately, abstraction techniques help to tackle the problem entailed by this restriction and can therefore cope with infinite colours. [16] proposes an algorithm to construct a finite state abstract program from a given, possibly infinite, state program (e.g., a CWF-net) by means of a syntactic program transformation starting with an initial set of predicates from a specification (e.g., modal specification).

[4] `http://projects.eclipse.org/projects/modeling.sirius`

[5] `https://sicstus.sics.se`

Table 3. Experimentation Results

#	Formula	φ	K	$\phi(K)$	Result
p_1	$QA \models_{must} [true]Register[u = u_1] \wedge .. \wedge Register[u = u_t]$	TRUE	-	-	TRUE
p_2	$QA \models_{may} [true]ViewA[r = admin]$	FALSE	-	-	FALSE
p_3	$QA \models_{may} [true]CreateQ[r = client, u = u_x]$ $\wedge RefuseA[r = client, u_1 = u_x]$	TRUE	5	FALSE	-
			7	TRUE	TRUE
p_4	$QA \models_{must} [true]CommitQ[u_2 = u_x] \Rightarrow CommitA[u_2 = u_x]$ $\wedge ArchiveQA[u_2 = u_x]$	TRUE	-	-	TRUE
p_5	$QA \models_{must} [true]\neg CreateA[r = client]$	TRUE	-	-	TRUE
p_6	$QA \models_{may} [nbUsers > 3]CreateQ[u = u_x] \wedge \neg CreateQ[u = u_y]$	TRUE	12	TRUE	TRUE
p_7	$QA \models_{must} [nbUsers < 3]\neg CreateQ[true]$	TRUE	-	-	TRUE
p_8	$QA \models_{must} [true]CreateQ[true] \Rightarrow (Register[r = client]$ $\wedge Register[r = seller] \wedge Register[r = admin]$	TRUE	-	-	TRUE

This method is shown to be sound (the abstract program is always guaranteed to simulate the original one) and complete (the algorithm can produce a finite simulation-equivalent, resp. bisimulation-equivalent, abstract program if the concrete program has a finite abstraction with respect to simulation, resp. bisimulation, equivalence). On the one hand, in the case of a bisimulation-equivalent abstract program, the abstracted modal specification can be verified using our method, and the (in)validity of the modal specification can be directly inferred. On the other hand, for simulation-equivalent abstract program, only the validity of a *may*-formula and the invalidity of a *must*-formula can be inferred.

To handle infinite colours, another approach is to consider only a finite number of data of an infinite colour according to control-flow selection criteria (e.g., decision or condition coverage) [17]. However, this approach is not complete.

5.2 Experimental Results

The approach and the corresponding implementation have been applied to the industrial issue tracking system described in Sect. 3. Since the properties have initially been defined by the business analysts involved in the project, we assume that they are representative of properties that should be verified by engineers when they design and implement such business processes. Furthermore, the obtained verification results have been shared and discussed with them. Table 3 shows an extract of the experimental results focusing on the properties p_1 to p_8 from Sect. 3. In Tab. 3, the modal formula associated with each property is specified, and the computation result is given by its final verdict (valid or not) as well as the internal evaluation of φ. The input K and the corresponding computed value of $\phi(K)$ are also precised when it makes sense, i.e. when the algorithm cannot conclude without this bound.

On the one hand, we observe that when verifying *must*-formulae that are satisfied by the CWF-net (e.g., p_1, p_4), or *may*-formulae that are not satisfied by the WF-net (e.g., p_2), the over-approximation φ is usually enough to conclude. On the other hand, when verifying *may*-formulae that are satisfied by the CWF-net (e.g., p_3), or *must*-formulae that are not satisfied by the WF-net, the decomposition into K segments is needed. We empirically show that this decomposition is very effective since values of K_{max} are usually moderate ($K_{max} = 12$ for p_6, less than 30 on all the experiments conducted on this case study).

Thanks to the experiments conducted using this proof-of-concept prototype, we can conclude that the proposed method is suitable and efficient, and can therefore gain benefits within business process design and verification. Notably, these experiments highlighted that the approach is able to conclude about the (in)validity of the studied properties in a very short time (less than 5 seconds).

6 Conclusion and Related Work

This paper presents an approach based on constraint systems to model executions of CWF-nets in order to verify modal specifications. It allows managing realistic and complex specifications that manipulate and manage data types. This approach, supported by an Eclipse-based prototype, has been successfully experimented on a non-trivial case study to validate an issue tracking system. These promising results show the relevance and the effectiveness of the approach to validate complex business processes using modal specifications.

Modal specifications–originally introduced in [9]–allow loose or partial specifications in a process algebraic framework. Adapted to Petri nets, they allow defining relations between generated modal languages to decide specifications refinement and asynchronous composition [18]. In [10], modal specifications language over WF-nets expresses requirements on several activities and on their causalities. To handle CWF-nets, we extend modal specifications with additional conditions on initial state as well as on coloured transitions.Unlike [18], to verify modal specifications, our approach focuses on correct executions of CWF-nets.

A lot of results have been provided to model and to analyse Petri nets by using equational approaches [19]. Among popular resolution techniques, constraint programming has been successfully used to analyse properties of Petri nets. In [20], an SMT-based approach to the coverability problem using the state equation and traps is presented. Our CSP-based approach also takes advantage of trap and siphon properties in pursuance of modelling correct executions of CWF-nets. Furthermore, constraint programming makes it possible to tackle one of the major verification problems–the reachability problem, as shown in [21] where a decomposition into *step sequences*, i.e. segments, was modelled by constraints. Our approach is almost similar, but the constraints on step sequences are much stronger in our case because we address not only the reachability of a given marking, but also the transitions involved in the path reaching it.

As a future work, we plan extensive experiment to increase the scalability of our verification approach based on constraint systems. To improve its readiness level and to foster its use by business analysts, we plan to propose user-friendly modal properties patterns. On the theoretical side, investigating modal specifications preservation through refinements is a further research direction.

References

1. Cardoso, J., Mendling, J., Neumann, G., Reijers, H.A.: A discourse on complexity of process models. In: 2^{nd} Workshop on Business Process Intelligence (BPI'06). Volume 4103 of LNCS., Vienna, Austria, Springer (September 2006) 117–128

2. Dumas, M., ter Hofstede, A.H.M.: UML Activity Diagrams As a Workflow Specification Language. In: 4^{th} Int. Conf. on The Unified Modeling Language (UML'01), Toronto, Canada, Springer-Verlag (October 2001) 76–90
3. van der Aalst, W.M.P.: Three Good reasons for Using a Petri-net-based Workflow Management System. Journal of Information and Process Integration in Enterprises **428** (December 1997) 161–182
4. van der Aalst, W.M.P., Van Hee, K.M., Houben, G.J.: Modelling and analysing workflow using a Petri-net based approach. In: Workshop on Computer-Supported Cooperative Work, Petri nets and related formalisms. (June 1994) 31–50
5. Ellis, C.A., Nutt, G.J.: Modeling and enactment of workflow systems. In: 14^{th} Int. Conf. on Application and Theory of Petri Nets (Petri Nets'93). Volume 691 of LNCS., Chicago, IL, USA, Springer (June 1993) 1–16
6. Liu, D., Wang, J., Chan, S.C.F., Sun, J., Zhang, L.: Modeling workflow processes with colored Petri nets. Computers in Industry **49**(3) (December 2002) 267–281
7. Larsen, K.G.: Modal Specifications. In: Int. Workshop on Automatic Verification Methods for Finite State Systems. Volume 407 of LNCS., Grenoble, France, Springer-Verlag (June 1989) 232–246
8. Jensen, K.: Coloured Petri nets. In: Petri Nets: Central Models and Their Properties. Volume 254 of LNCS. Springer (1987) 248–299
9. Larsen, K.G., Thomsen, B.: A modal process logic. In: 3^{rd} Annual Symp. on Logic in Computer Science (LICS'88), Edinburgh, UK, IEEE CSP (July 1988) 203–210
10. Bride, H., Kouchnarenko, O., Peureux, F.: Verifying modal workflow specifications using constraint solving. In: Int. Conf. on Integrated Formal Methods (IFM'14). Volume 8739 of LNCS., Bertinoro, Italy, Springer (September 2014) 171–186
11. Murata, T.: Petri nets: Properties, analysis and applications. IEEE **77**(4) (April 1989) 541–580
12. Presburger, M.: Über die vollständigkeit eines gewissen systems der arithmetik ganzer zahlen, in welchem die addition als einzige operation hervortritt. In: Sprawozdanie z I Kongresu metematykw slowiaskich, Warszawa, Poland (1929) 92–101
13. Macworth, A.K.: Consistency in networks of relations. Journal of Artificial Intelligence **8**(1) (1977) 99–118
14. van Hentenryck, P., Dincbas, M.: Domains in logic programming. In: Nat. Conf. on Artificial Intelligence (AAAI'86). (August 1986) 759–765
15. Minsky, M.L.: Computation: finite and infinite machines. Prentice-Hall, Inc. (1967)
16. Namjoshi, K.S., Kurshan, R.P.: Syntactic program transformations for automatic abstraction. In: 12^{th} Int. Conf. on Computer Aided Verification (CAV'00). Volume 1855 of LNCS., Chicago, IL, USA, Springer (July 2000) 435–449
17. Vilkomir, S., Bowen, J.: Formalization of software testing criteria using the Z notation. In: 25^{th} Int. Conf. on Computer Software and Applications (COMPSAC'01), Chicago, IL, USA, IEEE CSP (October 2001) 351–356
18. Elhog-Benzina, D., Haddad, S., Hennicker, R.: Refinement and asynchronous composition of modal Petri nets. In: Transactions on Petri Nets and Other Models of Concurrency V. Volume 6900 of LNCS. Springer (2012) 96–120
19. Desel, J.: Basic linear algebraic techniques for place/transition nets. In: Lectures on Petri Nets I: Basic Models. Volume 1491 of LNCS. Springer (1998) 257–308
20. Esparza, J., Ledesma-Garza, R., Majumdar, R., Meyer, P., Niksic, F.: An SMT-based approach to coverability analysis. In: 26^{th} Int. Conf. on Computer Aided Verification (CAV'14), Vienna, Austria, Springer (2014) 603–619
21. Bourdeaud'huy, T., Hanafi, S., Yim, P.: Incremental Integer Linear Programming Models for Petri Nets Reachability Problems. Petri Net: Theory and Applications (February 2008) 401–434