Positron: Composing Commitment Protocols

Paper 13

No Institute Given

Abstract. We consider *(commitment) protocols* specified formally in terms of their roles, their messages, and the meanings of their messages (expressed as commitments). In an important advance over previous work, we show how to *compose* protocols, thereby facilitating reuse. We address two role-specific aspects of composition: (1) *role requirements*, capturing the benefits a role receives from the composite protocol; and (2) *role accountability*, capturing the commitments a role makes to other roles—to promote their joint enactments of the composite protocol. Our approach yields benefits in business requirements elicitation (natural abstraction); enactment (flexibility); and compliance and validation (ascribing accountability for each requirement to a specific role). We evaluate our contributions by modeling well-known protocols in the insurance, manufacturing, and healthcare domains.

Keywords: Commitments, Agent communication, Verification of multiagent systems, Communication protocols, Model checking

1 Introduction

We adopt an interaction-oriented stance on multiagent systems, e.g., as applied in cross-organizational service engagements. We consider *(commitment) protocols*, which specify the interactions between two or more roles in terms of how their messages relate to their *commitments* [24], obtaining well-recognized benefits in dealing with the autonomy and heterogeneity of business partners [2, 30].

Composition is a key construct in software engineering. We address two role-specific aspects of composition: *role requirements* which capture the benefits a role receives from the composite and *role accountability* which captures the commitments a role must make to other roles.

1.1 Real-Life Scenario: AGFIL

We illustrate our approach using the real-life, automobile insurance claims processing case for AGF Irish Life Holding (AGFIL) [3]. As Figure 1 illustrates, this case involves four parties along with POLICYHOLDER and ADJUSTER (not shown). AGFIL underwrites automobile insurance policies and covers losses incurred by policy holders. Europ Assist (EA) provides a 24-hour help-line service for receiving claims. Approved REPAIRERS provide repair services. Lee Consulting Services (Lee) coordinates with AGFIL, repairers, and adjusters to handle a claim.

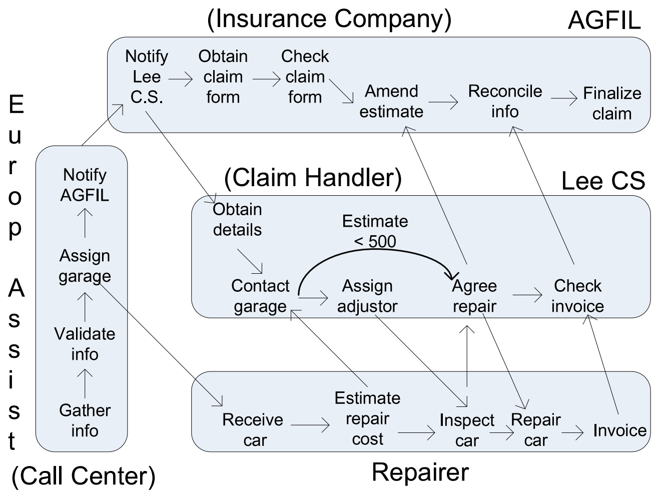

Fig. 1. Traditional model of a cross organizational insurance claim processing [3].

The traditional model of the AGFIL scenario describes the workflows of each partner along with how they relate to one another. Such a description, even if supported by standards such as BPMN [20], tightly couples the inner workings of the partners. Newer approaches deemphasize the inner workings and instead capture the interactions between the business partners more explicitly via a formal notation [12, 23, 28]. These approaches express constraints on the ordering and occurrence of the messages exchanged by the business partners.

In contrast, a commitment protocol emphasizes the social state of an interaction, expressed here in terms of commitments. A protocol describes the roles involved, the messages they exchange, and any preconditions on and effects of the messages on the social state. An agent adopts a role and enacts the specified protocol by autonomously choosing (in accordance with its internal policies) how to interact.

1.2 Contributions and Organization

Although protocols offer significant benefits over traditional approaches, protocols are not fully viable for the following reasons. One, specifying in one shot an adequate protocol for a complex scenario is nontrivial. Two, implementing agents who can play roles in such a comprehensive protocol is difficult because the differing details of the protocols complicate reusing parts of agent implementations. Our contribution in this paper is to show how complex protocols can be constructed by composing existing protocols. Previous relevant research falls into these categories: (a) commitments but not composition [10]; (b) composition but no commitments [19, 25]; and (c) composition and commitments. The last category can be categorized as (c1) purely abstract description without a specification language or tools [15]; (c2) composition of commitment-based protocols based on regulative constraints [16]; and (c3) our approach to composition of commitment-based protocols based on role responsibilities and accountabilities.

Our approach, *Positron*, extends the Proton approach [10] to provide a clear syntax and semantics for composite protocols. Where Proton checks protocol refinement, Positron composes protocols. Positron (a) recursively expands nested constituent protocols; (b) introduces *composite protocol diagrams* as a graphical notation, conveying important features of the composite protocol to business and technical stakeholders; (c) introduces *role requirements* and *role accountabilities*; (d) incorporates a methodology for composing commitment protocols; and (e) implements a decision procedure and mechanical verification of protocols with respect to role requirements, role accountabilities, and enactments, compiling formulas to temporal logic, and employing MCMAS [13], a leading model checker, to verify if the composite protocol satisfies those formulas.

We describe relevant background, the major elements of our technical approach, and our methodology for constructing composite protocols. We evaluate our approach by modeling protocols from the insurance, manufacturing, and healthcare domains, that other researchers have studied, and summarize our results and experiences. We conclude with a discussion of the relevant literature and future work.

2 Background

We formulate a commitment [24] as being *from* a set of debtors *to* a set of creditors that if the antecedent begins to hold, the debtors will bring about the consequent in the future. In symbols: $\mathsf{C}_{\{\text{debtors}\},\{\text{creditors}\}}(\textit{antecedent}, \textit{consequent})$ [10]. Both antecedent and consequent are Boolean expressions over state-space propositions. When the antecedent becomes true, the commitment is *detached*, and the debtors become *unconditionally* committed to the creditors. When the consequent becomes true, the commitment is *discharged*. Debtors *should* discharge their detached commitments. However, debtors are autonomous and may violate a commitment by canceling it. The only compliance requirement for commitments is: each detached commitment *must* eventually be discharged (satisfied, transferred or released) or canceled. Specifically, there is no implicit ordering constraint between the antecedent and consequent.

3 Technical Approach

Positron provides a formal language in which to express composite protocols based on existing constituent protocols. (We lack the space to present the detailed syntax, but present an example in Listing 1 later.) Recall that Proton provides a language for capturing roles, propositions, and messages with guards on when they can be sent, and their effects on the commitments of roles [10]. Positron augments the Proton language by adding constructs to define a composite protocol *using* a set of parameterized constituent protocols and defines a protocol composition methodology.

Further, while it accepts and verifies any CTL expression, Positron introduces five constructs for common verification patterns when composing protocols: *Req* function for role requirements, *coupling commitments* for role accountabilities, and three *path expressions* for good and bad enactments.

3.1 Protocol Composition

Positron supports nested composition of protocols. A composite protocol P can use (include) an instance of a constituent protocol with a `use` statement $q : Q(\overline{x} = \overline{p})$ specifying instance name (q), protocol type (Q), a set of arguments $\overline{p}$ passed by P, and a matching set a parameters $\overline{x}$ accepted by constituent Q. Arguments and parameters are named and have a type of either role or proposition. The argument and parameter sets must contain matching names and types. Positron expands P to produce a new, flatter protocol P'. Expansion gives every element in Q a new, unique name, and replaces each parameter with its corresponding argument. Unique names are constructed by preappending the instance name q to each element name in Q.

Definition 1. *Given a set of arguments $\overline{p}$, a parameterized constituent protocol instance Q that accepts a set of parameters $\overline{x}$, and the sets $\overline{p}$ and $\overline{x}$ agree in both name and type. Define $Q^{\overline{x}}_{\overline{p}}$ as Q in which all elements in Q are given unique names, and every parameter in $\overline{x}$ is replaced with its corresponding argument in $\overline{p}$.*

Protocol expansion of a composite P containing multiple constituent instances $q : Q(\overline{x} = \overline{p})$ is the union of P and $Q^{\overline{x}}_{\overline{p}}$, and removing the use statement $Q^{\overline{x}}_{\overline{p}}$ from P's uses. The defintion expands any one contituent. Apply the defintion repeatedly to expand all constituent instances.

Definition 2. *Given a composite protocol P that uses multiple constituent protocol instances $q : Q(\overline{x} = \overline{p})$, where P passes a set of arguments $\overline{p}$, Q accepts a set of parameters $\overline{x}$, and the sets $\overline{p}$ and $\overline{x}$ agree in name and type. Then protocol $P' = \mathit{expand}(P,\ q : Q(\overline{x} = \overline{p})$ is the expanded version of P and Q, and is defined as*

$$\mathit{roles}(P') := \mathit{roles}(P) \cup \mathit{role}(Q^{\overline{x}}_{\overline{p}}) \tag{1}$$

$$\mathit{props}(P') := \mathit{props}(P) \cup \mathit{props}(Q^{\overline{x}}_{\overline{p}}) \tag{2}$$

$$\mathit{commitments}(P') := \mathit{commitments}(P) \cup \mathit{commitments}(Q^{\overline{x}}_{\overline{p}}) \tag{3}$$

$$\mathit{messages}(P') := \mathit{messages}(P) \cup \mathit{messages}(Q^{\overline{x}}_{\overline{p}}) \tag{4}$$

$$\mathit{checks}(P') := \mathit{checks}(P) \cup \mathit{checks}(Q^{\overline{x}}_{\overline{p}}) \tag{5}$$

$$\mathit{uses}(P') := (\mathit{uses}(P) - q) \cup \mathit{uses}(Q^{\overline{x}}_{\overline{p}}) \tag{6}$$

where $\mathit{roles}(Q)$, $\mathit{props}(Q)$, $\mathit{commitments}(Q)$, $\mathit{messages}(Q)$, $\mathit{checks}(Q)$, and $\mathit{uses}(Q)$ refer to the corresponding element sets of protocol Q.

3.2 Role Requirements

Role requirements are the requirements that an agent playing a role places on the composite protocol. A designer specifies a role requirement in the Positron language, which Positron compiles into a CTL formula. As an example, in the AGFIL scenario, POLICYHOLDER expect his car will be repaired if he has an accident. Positron could compile this requirement into the CTL specification: $\mathbf{AG}(\mathit{accident} \rightarrow \mathbf{AF}\ \mathit{repair})$.

However, such a requirement ignores *business exceptions*: a commitment may fail because its debtor either chooses not to, or is prevented by circumstances from, discharging it. In verifying a role requirement, we cannot assume commitments are never canceled. Rather, we state role R's requirement as: if R fulfills all its commitments and p holds at any state, then always eventually, either q holds or a role other than R cancels one of its commitments. If R's requirement fails because R cancels a commitment, that is not a fault of the protocol, but of R. In CTL, where $r.\mathit{anyCancel}$ is true if and only if role r cancels any of its commitments, this is

$$\mathit{Req}(R,p,q) \triangleq \mathbf{AG}(p \rightarrow \mathbf{AF}(q \vee \bigvee_{r \neq R} r.\mathit{anyCancel}))$$

In AGFIL, one of POLICYHOLDER's role requirements is captured as: if INSURER offers coverage, I paid the premium, and I have an accident, then my car will be repaired: $\mathit{Req}(\mathrm{PH}, \mathit{coverage} \wedge \mathit{premium} \wedge \mathit{accident}, \mathit{repair})$

3.3 Enactment Requirements

Although capturing all possible enactments is not feasible, designers often know of specific good and bad enactments. We use the specified *enactments* as bases for verifying a composite protocol to assist designers in refining the protocol specification (e.g., its constituent protocols and coupling commitments) or the requirements. Our notion of enactments resembles scenarios from scenario-based requirements engineering [9]. In essence, each enactment corresponds to a unit test in software engineering.

We use model checking to verify enactments. We introduce three recursive functions to simplify enactment specification. Given an enactment P, which is an ordered list of Boolean expressions over states and messages, where $\mathit{head}(P)$ is the first element in list P, and $\mathit{tail}(P)$ is P without the first element, let

$$\mathit{EXPath}(P) \triangleq \begin{cases} \mathit{head}(P) \wedge \mathbf{EX}(\mathit{EXPath}(\mathit{tail}(P))) & \text{if } |P| > 1 \\ \mathbf{EX}(P) & \text{if } |P| = 1 \end{cases}$$

$$\mathit{EFPath}(P) \triangleq \begin{cases} \mathbf{EF}(\mathit{head}(P) \wedge \mathit{EFPath}(\mathit{tail}(P))) & \text{if } |P| > 1 \\ \mathbf{EF}(P) & \text{if } |P| = 1 \end{cases}$$

$$\mathit{EUPath}(r,P) \triangleq \begin{cases} \mathbf{E}(\neg r \ \mathbf{U} \ (\mathit{head}(P) \wedge \mathit{EUPath}(r, \mathit{tail}(P)))) & \text{if } |P| > 1 \\ \mathbf{E}(\neg r \ \mathbf{U} \ P) & \text{if } |P| = 1 \end{cases}$$

EXPath: specifies a path of states, beginning at a start state, that must appear consecutively without skipping over other states. In protocols with many constituents, *EXPath* can be too strong a constraint, since it precludes interleaving of constituent protocols.

EFPath: specifies a path of states that must appear in order, but allows other states to be interleaved. This is a weaker constraint than *EXPath*.

EUPath: specifies a path of states that must appear in order, and constrains which states can be interleaved in the path. Expression r identifies which states must *not* be interleaved in the path. An *EUPath* constraint is stronger than *EFPath* and weaker than *EXPath*.

Two enactments from AGFIL are

$$EFPath(accident, deliverReq, deliverCar, \ldots, repair)$$
$$\neg EFPath(repair, accident)$$

3.4 Coupling Commitments

The constituent protocols occurring in a (nontrivial) composite protocol must be interrelated. In a multiagent system, some role must be accountable for ensuring constituent protocols are properly interrelated. We capture the *role accountability* implied by such an interrelationship via a *coupling commitment*. A coupling commitment's debtor is the accountable role, and its creditors are (in general) the union of all roles connected by the interrelated constituent protocols, minus the debtor. Like any commitment, debtor commits to discharge consequent if antecedent becomes true.

$$\mathsf{C}_{\text{accountable role}, \{\text{interrelated roles}\}}(antecedent, consequent)$$

Two coupling commitments from AGFIL are

$$\mathsf{C}_{CC, \{PH,\ Re\}}(deliverReq, notify\textsc{Re})$$
$$\mathsf{C}_{PH, \{CC, Re\}}\ (deliverReq \wedge approval, deliverCar)$$

3.5 Verification

Positron reads designer written source code for the composite and constituent protocols and generates a single MCMAS input file. MCMAS reads the input, builds the model, and reports whether each CTL formula holds in the model.

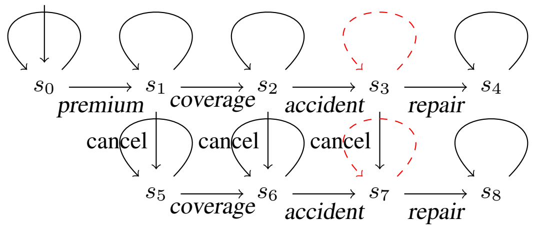

Fig. 2. Selected states and transitions for AGFIL.

Figure 2 shows a portion of the state space Positron generates for verification from AGFIL's constituent protocols and coupling commitments. The start state is denoted by the unlabeled line in the top left. Solid black lines are valid transitions (messages); dashed red lines are invalid transitions. Since the message guard for *coverage* is *premium*, *coverage* can occur only after *premium*, making s_1 an invalid start state. The other states are also invalid start states.

Notice that the top row (*premium*, *coverage*, *accident*, and *repair*) begins a good enactment. Positron can verify the existence of this path using *EFPath* requirement. Further, Positron can ensure that the model is free of specific bad enactments, for example, that s_1 must not be a start state.

To capture the only commitment compliance requirement, Positron generates a model checking fairness constraint for each commitment: a commitment must not remain unconditional and unresolved forever. Red dashed loops are invalid because they violate a fairness constraint.

A composite protocol may fail to satisfy role or enactment requirements for different reasons.

Ver_{RR}: If a role requirement formula based on the Req function fails, then coupling commitments are missing; add coupling commitments that require agents to act.

Ver_{G}: If a good enactment formula fails, then either (Ver_{GG}) some message guards are too strong; weaken guards to validate additional good transitions. Or (Ver_{GC}) some commitment become detached, but can never resolve; weaken guards to validate transitions that satisfy the commitment's consequent.

Ver_{B}: If a bad enactment formula fails, then some message guards are too weak; strengthen guards to invalidate existing incorrect transitions.

3.6 Composite Protocol Diagrams

We propose *composite protocol diagrams* (CPDs) as a notation that displays the essence of a composite protocol both to business analysts and technical designers, who collaborate in its construction. We use CPDs in this paper to help visualize a large amount of protocol information, but we defer evaluation of this visual notation to future work. CPD diagrams focus designers' attention by summarizing high-level business relationships between the roles as reusable constituent protocols, hiding the details of each constituent.

Figure 3 shows the CPD for composite protocol AGFIL. It contains constituent protocol instance PH-IN of type *Exchange*. PH $\xrightarrow{R1}$ PH-IN shows role PH enacts role *R1* in constituent protocol PH-IN. The two unlabeled circular arcs centered on POLICYHOLDER represent coupling commitments, implicitly named PH_1 (inner) and PH_2 (outer). (role names REQ and R1 label straight edges, not arcs.)

4 Methodology

This section describes, and Table 1 summarizes, our iterative methodology to develop a CPD such as that of Figure 3.

Step 1 (Roles): Identify all roles with a business function, i.e., that potentially enter into commitments with others.

Step 2 (Constituent Selection): Identify all business relationships among different subsets of roles and identify constituent protocols that realize such relationships. Examine message flows since they suggest the presence and nature of the constituent protocols. If a suitable protocol is known, use it; else apply the methodology recursively. The constituent name is the protocol name in the `use` statement of the Positron source.

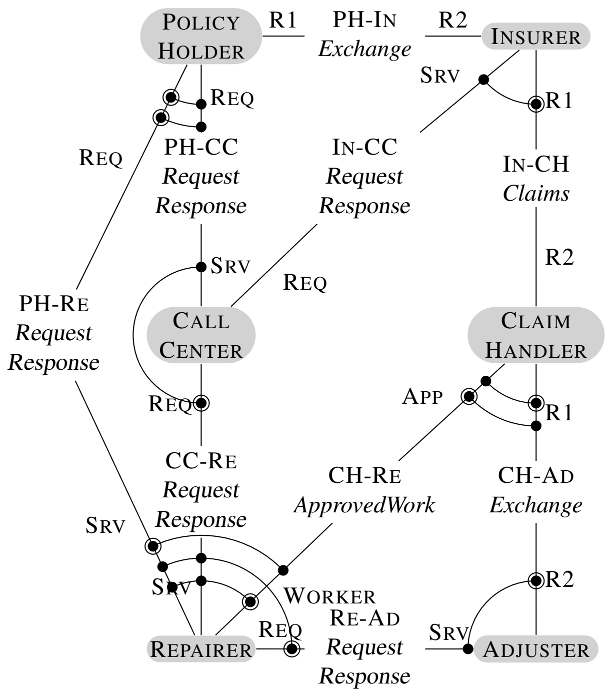

Fig. 3. Composite protocol diagram for AGFIL. A CPD for a *composite* protocol displays its contained *constituent* protocol instances (unshaded labeled nodes). *Composite roles* (shaded labeled nodes) and constituents are connected by *constituent roles* (straight labeled edges). Unlabeled circular arcs represent coupling commitments with input constituents (dot) and output constituents (target symbol).

Step 3 (Role Requirements): Specify each role's requirements as Req functions.

Step 4 (Enactments): Incrementally specify good and bad enactments. The enactments can be developed by a role-playing process similar to that described by Parunak [22]. They should cover representative enactments that support or invalidate each of the role requirements identified in the previous step. Manually tracing enactments throughout a model is tedious and error prone. Explicitly specifying enactments enables the model checker to trace the enactments mechanically, after each incremental composite change.

Step 5 (Coupling Commitments): Examine each good enactment from beginning to end, and assume roles do only what is *minimally* required to discharge their commitments. When adjacent steps of a good enactment are messages from the same constituent protocol, verify that the constituent accurately enacts those steps, or select a different constituent. When adjacent steps of an enactment are messages from differ-

Table 1. Inputs and outputs for each step of the methodology.

Step	Name	Inputs	Outputs
1	Roles	Background and requirements	Roles in the composite
2	Constituent Selection	Role relationships and protocol library to be composed	Constituent protocols
3	Role Requirements	Role business needs	Role requirements
4	Enactments	Background knowledge of requirements	Good and bad enactments
5	Coupling Commitments	Enactments complete CPD	Coupling commitments;
6	Positron	All artifacts	Positron source code
7	Verification	Positron source code	Model checker results

ent protocols (where some role receives a message in one constituent and then sends a message in another constituent), add a coupling commitment for that role.

Step 6 (Positron): Given the previous methodology artifacts, write the Positron source code, including roles, constituent protocols, role requirements, coupling commitments and enactments.

Step 7 (Verification): Run Positron and MCMAS to verify all formulas for role and enactment requirements.

5 Evaluation

We evaluate our contributions by modeling protocols from the insurance, manufacturing, and healthcare domains.

5.1 AGFIL Evaluation

We extend the AGFIL protocol by adding (a) POLICYHOLDER and accident reporting; (b) ADJUSTER and the redirection of two messages between CLAIMHANDLER and REPAIRER through ADJUSTER; (c) payments from REPAIRER to CLAIMHANDLER to INSURER; (d) a protocol for premiums and coverage between POLICYHOLDER and INSURER; and (e) REPAIRER returning the car.

At the top of Figure 3, POLICYHOLDER (enacting role R1) purchases insurance from INSURER (enacting role R2) using an instance of constituent protocol *Exchange* named PH-IN. POLICYHOLDER reports accidents to CALLCENTER using an instance of constituent protocol *RequestResponse* named PH-CC. CALLCENTER notifies INSURER (IN-CC), and assigns and notifies REPAIRER (CC-RE). INSURER passes the claim to CLAIMHANDLER (IN-CH). POLICYHOLDER and REPAIRER exchange the damaged and later repaired car (PH-RE). CLAIMHANDLER, REPAIRER, and ADJUSTER inspect the car and approve repairs (CH-AD, RE-AD and CH-RE).

Step 1 (Roles): Figure 1 refers to specific agents (companies), not roles. Declare six roles: role INSURER (abbreviated IN) for agent AGFIL, CALLCENTER (CC) for

EUROP ASSIST, and CLAIMHANDLER (CH) for LEE. The other roles are POLICYHOLDER (PH), REPAIRER (RE), and ADJUSTER (AD). At the end of this step, the CPD diagram in Figure 3 shows only the six shaded role nodes.

Step 2 (Constituent Selection): Assume protocols *RequestResponse* and *Exchange* (where two roles swap items) already exist. Designers recursively create *Claims* for IN-CH and *ApprovedWork* for CH-RE. Designers add the constituent protocol nodes and edges to Figure 3, completing all CPD nodes and edges.

Step 3 (Role Requirements): POLICYHOLDER requires: (1) if he has coverage, pays his premium, and has an accident, his car is repaired; (2) if he delivers his car to REPAIRER, his car is returned. INSURER requires: if a claim is filed, the claim is finalized. All roles except POLICYHOLDER require payment if they perform their tasks. All these are described as Req functions. Role requirements can *also* be specified directly in CTL, e.g., INSURER requires no car repairs without an inspection: $AG(\neg(\mathit{repair} \wedge \neg \mathit{inspectCH}))$.

Step 4 (Enactments): An important good enactment is that of reporting an accident and getting car repaired: (a) POLICYHOLDER reports an accident to CALLCENTER (PH-CC); (b) CALLCENTER assigns and notifies REPAIRER to repair the car (CC-RE); (c) CALLCENTER asks POLICYHOLDER to deliver his car to a specific REPAIRER (PH-CC); (d) POLICYHOLDER delivers car to REPAIRER (PH-RE); Remaining steps are omitted. Performing repairs before an accident is reported is a bad enactment: (e) car repaired; (f) accident reported.

Step 5 (Coupling Commitments): Between messages (a) and (b) of the accident-reporting enactment (see previous step), if POLICYHOLDER reports an accident, CALLCENTER assigns and notifies REPAIRER: $\mathsf{C}_{\mathrm{CC},\{\mathrm{PH},\mathrm{RE}\}}(\mathit{accident}, \mathit{notify}\mathrm{RE})$ Between messages (c) and (d), if CALLCENTER asks POLICYHOLDER to deliver his car to REPAIRER, he does so.

$\mathrm{PH}_1 = \mathsf{C}_{\mathrm{PH},\{\mathrm{CC},\mathrm{RE}\}}(\mathit{deliverReq} \wedge \mathit{approval}, \mathit{deliverCar})$

Adding arcs, the complete AGFIL CPD is Figure 3.

Step 6 (Positron): Listing 1 shows some lines from the Positron source file for AGFIL. Lines 2-3 declares all roles and propositions. Line 4 instantiates an instance of *Claims* named IN-CH. Lines 10 and 11 are two coupling commitments. Line 14 lists one of POLICYHOLDER's role requirements. Line 15 lists an INSURER requirement as explicit CTL. Line 17 verifies the good, accident-reporting enactment that must exist in the composite, and Line 18 verifies a bad enactment that must not exist.

The conversion of Positron to MCMAS maps role to role; proposition to boolean; commitment to enum and fairness condition; message to action; and formula expansion to formula. It also defines proposition and commitment evolutions, maps high-level Positron expressions to low-level MCMAS expressions, tracks each agent's last action and anyCancel values, and generates a large number of proposition and commitment state evaluation statements.

Step 7 (Verification): Running Positron and MCMAS successfully verified nine CTL formula: eight role and one enactment requirements. Removing any single coupling commitment caused one or more formulas to fail.

Listing 1 Positron source for AGFIL

```
 1: protocol AGFIL {
 2:     role $PH$; $In$; $CC$; $CH$; $Re$; $Ad$;
 3:     prop $accident$; $deliver$; $repair$; $paid$; ...
 4:     use $\text{IN-CH}$ : $Claims($
 5:         role $R1 = \text{IN}$, role $R2=\text{CH}$,
 6:         prop $pre1 = true$, prop $pre2=reportCH$,
 7:         prop $act1 = payCH$, prop $act2=repairIn)$;
 8:       ...
 9:     commitment
10:       $CC_1 : \mathsf{C}_{CC,\{PH,\,Re\}}(deliverReq, notify\text{RE})$;
11:       $PH_1 : \mathsf{C}_{PH,\{CC,Re\}}(deliverReq \wedge approval, deliver)$;
12:       ...
13:     formula
14:       $Req(\text{IN}, coverage \wedge premium \wedge accident, repair)$;
15:       $AG(\neg(repair \wedge \neg inspectCH))$;
16:       ...
17:       $EFPath(accident, deliverReq, \ldots, repair)$;
18:       $\neg EFPath(repair, accident)$;
19:       ...
20: }
```

6 Results and Experience

To demonstrate the broad applicability of Positron, our methodology was successfully able to create composite protocols for scenarios from three different business domains: AGFIL from insurance; Quote To Cash, an important business process for manufacturing supply chains [21]; and ASPE, a healthcare process for breast cancer diagnosis [1]. Positron successfully verified all role and enactment requirements. Table 2 shows statistics and timings for these three protocols.

Model Verification: We encountered and fixed a number of verification failures. Good enactment failures (Ver_{GG} and Ver_{GC}) were generally easier to fix, since they only require that some path exist. Bad enactment failures (Ver_{B}) require the impossibility of a particular enactment.

Model Validation: Identifying requirements was straightforward, but some initial specifications were incorrect because preconditions were missed. For example, POLICYHOLDER's role requirement on Line 14 initially failed (Ver_{RR}) because coupling commitment PH_1 on Line 11 did not include *approval*; REPAIRER will not always repair a car just because it is delivered to him. And, an accident is insufficient to get POLICYHOLDER's car is repaired; POLICYHOLDER must also have a policy and pay the premium.

Positron generates model checking fairness conditions to ensure all unconditional commitments eventually resolve. Initially, AGFIL's good enactment on Line 17 mysteriously failed because, even though the model allowed all the transitions, the good enactment had unresolvable commitments (invalid by commitment fairness conditions). We corrected the model so all commitments could resolve.

Table 2. Positron statistics. (M is 10^6 and G is 10^9.)

Composite Metric	**AGFIL**	**QTC**	**ASPE**
Constituent instances	11	6	12
Roles	6	6	5
Propositions	22	37	18
Commitments (all)	24	43	12
Coupling commitments	9	21	2
Messages	22	55	20
CTL Formulas	9	17	14
Role requirements	8	13	7
Enactment requirements	1	4	7
Positron statements	94	164	81
State space size	120M	381G	1.47M
Positron processing time	1.98s	3.16s	1.68s
MCMAS processing time	4.29s	1274s	5.78s
Total time	6.27s	1278s	7.46s

7 Discussion

Positron gains an advantage over traditional approaches by focusing on high-level business relationships realized as constituent protocols, and by focusing on commitments rather than control flow. Because role accountabilities are stated as commitments, if a requirement fails, we can trace the failure back to a specific role.

CPDs summarize relevant details about a composite protocol and we expect they will prove valuable, because they bring together both technical and business descriptions of protocols, helping bridge the Business-IT Divide [26].

7.1 Relevant Literature

Table 3 compares Positron with other work. Some papers propose a protocol specification language, and some propose an accompanying protocol specification methodology. Some papers address single protocols in isolation; some address common patterns within protocols; some address the composition of multiple protocols to create new composite protocols. Of those papers that address verification, some address business level requirements; some address verification properties between two protocols or models (such as protocol refinement); some address protocol-wide properties; some verify properties that must hold between the constituents of a composite protocol; some formulate role-specific properties; some formulate good or bad enactment properties; and some address other verification topics not addressed above.

Desai et al. [6] propose OWL-P [7, 5] and MAD-P [8] for specifying and verifying commitment protocols and their compositions. They employ axioms to specify a composition. These approaches suffer from a key drawback: an agent enacting the composite protocol may ignore the composition axioms leading to undesirable executions. In contrast, Positron employs coupling commitments with clear role accountability for

Table 3. Approach comparison. Column abbreviations and citations are Po=Positron; Pr=Proton; DA, DO, Dv and DM=Desai et al.; T=Telang and Singh; Y=Yolum; Mi=Miller and McBurney; G=Günay et al.; C=Cheong and Winikoff, Mc=McGinnis and Robertson, L=Lomuscio et al.., Ma=Marengo. Check marks show the significant topics addressed by each paper. The cell contents of the verification rows indicates whether the paper discusses (D) or mechanizes (M) verification.

Significant Topics	**Po**	**Pr**	**DA**	**DO**	**Dv**	**DM**	**T**	**Y**	**Mi**	**G**	**C**	**Mc**	**L**	**Ma**
Verification Topics		[10]	[6]	[7]	[5]	[8]	[27]	[29]	[18]	[11]	[4]	[17]	[14]	[16]
Protocol specification	✓	✓	✓	✓	–	✓	✓	✓	✓	✓	✓	✓	–	✓
Methodology	✓	–	✓	✓	–	–	✓	–	–	✓	✓	–	–	–
Single protocol	–	✓	✓	✓	✓	–	–	✓	✓	✓	–	✓	✓	✓
Protocol patterns	–	–	–	–	–	–	✓	–	–	–	–	–	–	–
Protocol composition	✓	–	✓	✓	✓	✓	–	–	✓	–	✓	–	✓	✓
Requirements verification														
Business	M	–	–	–	M	–	M	–	–	D	–	–	M	M
Protocol-to-protocol	–	M	–	–	–	–	M	–	–	–	–	–	M	–
Protocol	M	–	–	–	M	–	–	M	M	D	D	–	–	M
Inter-constituent	M	–	–	–	–	M	–	–	–	–	–	–	–	–
Role-specific	M	–	–	–	–	–	–	–	–	D	–	–	M	–
Enactments	M	–	–	–	–	–	M	–	–	–	D	–	M	M
Other	M	M	–	–	M	M	M	M	M	D	–	–	M	M

the effects of one constituent protocol on others. Further, Amoeba is purely manual, whereas Positron incorporates mechanical verification. Adopting Amoeba's event ordering idea would add flexibility to our approach, but more granular parameterizations of constituents provides the same functionality.

Telang and Singh [27] (T&S) describe a methodology for business modeling that captures the commitments to be created among the parties by melding selected business patterns. In contrast, a protocol in Positron additionally specifies the messages and guards, and the protocols are first-class entities that retain their identity in the composite protocol, yielding improved modularity and modifiability. Most significantly, T&S's approach verifies if one implementation is sound with respect to the model. In contrast, Positron verifies if the model itself is sound.

Yolum [29] proposes generic correctness properties of commitment protocols for design-time verification, but does not address composite protocols. She considers generic properties, whereas we consider role-specific business requirements. It would be interesting to formulate Yolum's generic correctness properties in Positron to help improve protocol designs.

Miller and McBurney [18] (M&M) propose the $\mathcal{RASA}$ language based on propositional dynamic logic (PDL) to specify and compose protocols. $\mathcal{RASA}$'s preconditions, actions and postconditions correspond to Positron's guards, messages and meanings. Positron additionally incorporates role requirements, coupling commitments, and good and bad enactment paths, making Positron practically viable (intentionally omitted from $\mathcal{RASA}$). Theses are important in naturally describing business protocols, as we demonstrated above. Whereas M&M describe a custom reasoner, we rely on CTL semantics as realized in MCMAS.

Günay et al. [11] treat protocols as sets of commitments and propose automatically generating such sets from an agent's beliefs, goals, and capabilities. In contrast, we offer a semiautomatic approach where a tool helps designers compose existing protocols. Automatic generation is attractive but may not be feasible for complex settings, although a hybrid approach of developing atomic protocols mechanically and composite protocols with human assistance might be viable.

Cheong and Winikoff [4] describe the Hermes system for goal-oriented interaction. They focus on interaction-level goals, where we focus on role-level requirements and commitments. Their action sequence diagrams capture only good enactments.

McGinnis and Robertson [17] propose an approach in which an agent sends a protocol specification to other agents at runtime, as a way to accomplish dynamic, runtime, protocol adaptation. They remark that their approach lacks a way to prevent the agents from making an undesirable change to a protocol. If their protocols were augmented with commitments, Positron can help address this gap. For example, an agent may not remove a message from a protocol that brings about the consequent of a detached commitment. Why they describe rules for dynamically changing protocols, they do not address formal verification of interaction properties.

Lomuscio et al. [14] semiautomatically compile and verify contract-regulated service compositions. They use temporal-epistemic logic to check whether agents comply with their contracts using MCMAS (the same tool we employ). A crucial difference is that whereas Lomuscio et al. consider service compositions, we consider protocol compositions. Since a protocol has a footprint distributed across two or more roles, dealing with their compositions is inherently more subtle. In Table 3, this is classified as a kind of composition, and includes verification between actual behaviors and contractually correct behaviors (two "protocols").

Marengo [16] considers a related problem where protocols are composed (grafted). Marengo uses regulative specifications (constraints) using Linear Temporal Logic (LTL). We propose a methodology and use role responsibilities, role accountabilities, and path enactments using CTL. These idea sets are complementary and worthy of further study.

BPMN [20] is an industry standard notation for business processes. Unlike Positron, BPMN is only semiformal, and does not lend itself to formal verification. BPMN's emphasis on control flow results in rigid processes. Positron minimally constrains the participants by specifying the process in terms of commitments. Protocols are building blocks in Positron—a process is composed of protocols.

7.2 Future Work

The foregoing opens up useful directions for future work. At the theoretical level, treating the goals of the participants [11] is natural. At the practical level, generating enactments via tooling would be valuable. At the empirical level, evaluating the effectiveness of Positron (the approach and the tool) with professional developers on cross-organizational business processes would be necessary to promote the adoption of Positron by industry.

References

1. ASPE. The importance of radiology and pathology communication in the diagnosis and staging of cancer: Mammography as a case study, Nov. 2010. Office of the Assistant Secretary for Planning and Evaluation, U.S. Department of Health and Human Services; available at http://aspe.hhs.gov/sp/reports/2010/PathRad/index.shtml.
2. M. Baldoni, C. Baroglio, and E. Marengo. Behavior-oriented commitment-based protocols. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI)*, pages 137–142, 2010.
3. S. Browne and M. Kellett. Insurance (motor damage claims) scenario. Document D1.a, CrossFlow Consortium, 1999.
4. C. Cheong and M. P. Winikoff. Hermes: Designing flexible and robust agent interactions. In V. Dignum, editor, *Handbook of Research on Multi-Agent Systems: Semantics and Dynamics of Organizational Models*, chapter 5, pages 105–139. IGI Global, Hershey, PA, 2009.
5. N. Desai, Z. Cheng, A. K. Chopra, and M. P. Singh. Toward verification of commitment protocols and their compositions. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pages 33:1–33:3. ACM, 2007.
6. N. Desai, A. K. Chopra, and M. P. Singh. Amoeba: A methodology for modeling and evolution of cross-organizational business processes. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 19(2):6:1–6:45, Oct. 2009.
7. N. Desai, A. U. Mallya, A. K. Chopra, and M. P. Singh. Interaction protocols as design abstractions for business processes. *IEEE Transactions on Software Engineering*, 31(12):1015–1027, Dec. 2005.
8. N. Desai and M. P. Singh. A modular action description language for protocol composition. In *Proceedings of the 22nd Conference on Artificial Intelligence (AAAI)*, pages 962–967, Vancouver, July 2007. AAAI Press.
9. D. Filippidou. Designing with scenarios: A critical review of current research and practice. *Requirements Engineering*, 2(1):1–22, Mar. 1998.
10. S. N. Gerard and M. P. Singh. Formalizing and verifying protocol refinements. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2012. In press; available at http://www.csc.ncsu.edu/faculty/mpsingh/papers.
11. A. Günay, M. Winikoff, and P. Yolum. Commitment protocol generation. In *Proceedings of the 10th AAMAS Workshop on Declarative Agent Languages and Technologies (DALT)*, pages 51–66, 2012.
12. HL7. Health Level Seven, 2007. http://www.hl7.org.
13. A. Lomuscio, H. Qu, and F. Raimondi. MCMAS: A model checker for the verification of multi-agent systems. In *Proceedings of the International Conference on Computer Aided Verification*, LNCS, pages 682–688, 2009.
14. A. Lomuscio, H. Qu, and M. Solanki. Towards verifying contract regulated service composition. *Journal of Autonomous Agents and Multi-Agent Systems*, 24(3):345–373, May 2012.
15. A. U. Mallya and M. P. Singh. An algebra for commitment protocols. *Journal of Autonomous Agents and Multi-Agent Systems*, 14(2):143–163, Apr. 2007.
16. E. Marengo. *2CL Protocols: Interaction Patterns Specification in Commitment Protocols*. PhD thesis, Universitá Degli Studi di Torino, Feb 2013.
17. J. McGinnis and D. Robertson. Dynamic and distributed interaction protocols. In *Adaptive Agents and Multi-Agent Systems*, volume 3394 of *Lecture Notes in Computer Science*, pages 167–184. Springer, 2005.
18. T. Miller and P. McBurney. Propositional dynamic logic for reasoning about first-class agent interaction protocols. *Computational Intelligence*, 27(3):422–457, 2011.

19. T. Miller and J. McGinnis. Amongst first-class protocols. In *Proceedings of the 8th International Workshop on Engineering Societies in the Agents World (ESAW 2007)*, volume 4995 of *LNCS*, pages 208–223. Springer, 2008.
20. OMG. Business process model and notation (BPMN), version 2.0 beta, June 2010. Object Management Group. http://bpmn.org/.
21. Oracle. Automating the Quote-to-Cash process, 2009. http://www.oracle.com/us/industries/045546.pdf.
22. H. V. D. Parunak. Visualizing agent conversations: Using enhanced Dooley graphs for agent design and analysis. In *Proceedings of the 2nd International Conference on Multiagent Systems*, pages 275–282, Kyoto, 1996. AAAI Press.
23. RosettaNet. Home page, 2009. http://www.rosettanet.org.
24. M. P. Singh. An ontology for commitments in multiagent systems: Toward a unification of normative concepts. *Artificial Intelligence and Law*, 7(1):97–113, Mar. 1999.
25. M. P. Singh. Information-driven interaction-oriented programming: BSPL, the Blindingly Simple Protocol Language. In *Proceedings of the 10th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 491–498, Taipei, May 2011. IFAAMAS.
26. H. Smith and P. Fingar. *Business Process Management: The Third Wave*. Megan-Kiffer Press, Tampa, 2002.
27. P. R. Telang and M. P. Singh. Specifying and verifying cross-organizational business models: An agent-oriented approach. *IEEE Transactions on Services Computing*, 5(3):305–318, July 2012.
28. WS-CDL. Web services choreography description language version 1.0, Nov. 2005. http://www.w3.org/TR/ws-cdl-10/.
29. P. Yolum. Design time analysis of multiagent protocols. *Data and Knowledge Engineering Journal*, 63:137–154, 2007.
30. P. Yolum and M. P. Singh. Commitment machines. In *Proceedings of the 8th International Workshop on Agent Theories, Architectures, and Languages (ATAL 2001)*, volume 2333 of *LNAI*, pages 235–247, Seattle, 2002. Springer.