Positron: Composing Commitment-Based Protocols

Scott N. Gerard[1], Pankaj R. Telang[2], Anup K. Kalia[3], and Munindar P. Singh[3]

[1] IBM, Research Triangle Park, Durham, NC 27709, USA; sgerard@us.ibm.com
[2] Cisco, Research Triangle Park, Durham, NC 27709, USA; ptelang@cisco.com
[3] NC State University, Raleigh, NC 27695, USA; {akkalia,singh}@ncsu.edu

Abstract. We understand a sociotechnical system (STS) as a microsociety in which social entities interact about and via technical entities. A protocol specifies an STS by describing how its members collaborate by giving meaning to their interactions. We restrict ourselves to protocols that specify messages between roles in terms of how they create and affect commitments among the roles.
A key idea of our approach, Positron, is that a protocol specifies the *accountability* of one role to another in addition to the *requirements* from each role. Specifically, Positron incorporates role accountability and role requirements as two integral aspects of protocol composition. In this way, it seeks to promote collaboration in STSs through natural requirements elicitation; flexibility enactment; and compliance and validation (ascribing accountability for each requirement to a specific role). Positron maps composite protocols to the representations of a well-known model checker as a way to verify protocols to assist in their correct formulation. We evaluate Positron by demonstrating it on real-life protocols.

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

1 Introduction

We study sociotechnical systems (STSs) wherein autonomous parties interact about and through technical entities [29]. STSs arise in a variety of collaborative settings, including cross-organizational service engagements. A protocol specifies an STS in abstract terms by describing two or more roles and the messages those roles may exchange along with meanings of those messages [7]. Protocols arise commonly in business, e.g., RosettaNet [25], and healthcare, e.g., HL7 [16]. By bringing forth collaboration requirements, protocols separate implementations from interactions, thereby promoting the flexibility of autonomous collaborators, such as is needed in sociotechnical systems.

Existing protocol approaches [16, 25], however, standardize the message formats and operational constraints, but not the meanings of those messages, which are left informally stated. The past several years have seen the development of approaches that apply *commitments* [26] to specify the meanings of the messages in a protocol [3, 34], where protocol designers define meanings. The commitment-based approaches help deal with the autonomy and heterogeneity of participants and promote flexibility in interactions. The benefits of commitments for modeling service engagements are well established. For example, Telang and Singh [30] demonstrated an error in a published

RosettaNet guideline when modeled using commitments. Therefore, for brevity, we do not review the extensive literature on commitments here. Singh [28] provides a conceptual summary.

Challenge: Composition. Composition is a key construct in software engineering as a way to promote reuse and modularity. Existing approaches for protocols, whether operational or commitment-based, do not adequately support their composition, because they incorporate internal details or lack a formal semantics of interactions. Existing approaches, e.g., HL7 [16] and RosettaNet [25], provide atomic two-party protocols with the intent for them to be composed but do not support the composition as such and do not provide a construct by which two or more protocols could be composed. Thus the separation of interaction and implementation fails above the level of the atomic (predefined) protocols. Section 5 reviews the literature in detail. Suffice it to state here that previous relevant research falls into these categories: (a) composition but no commitments [22, 27]; (b) commitments but no composition [14]; and (c) composition and commitments. The last subcategory can be further refined as (c1) purely abstract description without a specification language or tools [19]; (c2) composition of commitment-based protocols based on axioms [9, 11] or regulative constraints [4] but without producing a composite protocol for further reuse and composition; and (c3) our present approach to composition of commitment-based protocols based on role responsibilities and accountabilities.

Motivating Research Questions and Contributions. How can we (1) formalize accountability, a crucial element of secure collaboration, as a basis for formal modeling and verification of STSs and (2) support composition of protocols to specify STSs? We address this question by restricting ourselves to composition and verification of commitment protocols, deferring representing other normative relationships [29] and their mapping to agent decision-making to future work.

Specifically, we propose *Positron*, a language and verification approach that supports composing commitment-based protocols and formally reasoning about them to verify desired properties. Positron (a) introduces *role requirements*, which capture a role's motivation, and *role accountability*, which captures the commitments a role makes to other creditor roles (that benefit from those commitments) as elements of a composite protocol specification; (b) shows how to recursively expand nested constituent protocols; (c) supports a methodology for composing commitment protocols; and (d) provides a decision procedure and mechanical verification of protocols with respect to role requirements, role accountabilities, and enactments. Positron compiles protocol specifications into MCMAS [17] models and checks protocols against temporal logic formulas. It then employs the MCMAS model checker to verify if the composite protocol satisfies those temporal formulas. The Positron verifier builds upon the Proton [14], verifier for commitment protocol refinement, expanding it to tackle protocol composition.

2 Background and Motivation

We write $\mathsf{C}_{\{\text{debtors}\},\{\text{creditors}\}}(\mathit{antecedent}, \mathit{consequent})$ [14] to denote a commitment *from* the specified debtors *to* the specified creditors that if the antecedent begins to hold, the debtors will bring about the consequent. The antecedent and consequent are Boolean

expressions. For example, $C_{PH,\{CC,Re\}}$ $(deliverReq \wedge approval, deliverCar)$ denotes that POLICYHOLDER (PH) commits to CALLCENTER (CC) and REPAIRER (Re) that he will deliver his car to REPAIRER whenever requested and the repair request is approved.

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, for simplicity, by canceling it. The only computational requirement for commitments is: each detached commitment *must* eventually be discharged (satisfied, delegate, assigned, or released) or canceled. A commitment imposes no ordering constraint between the antecedent and consequent, although in specific settings there may be a practical constraint.

Running Example: AGFIL. The following real-life case involves automobile insurance claims processing for AGF Irish Life Holding (AGFIL) [5], as Fig. 1 summarizes. This case involves four parties plus 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.

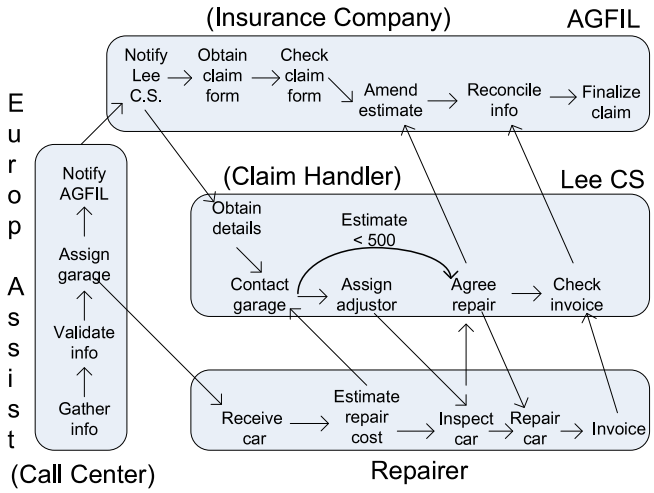

Fig. 1. Traditional process model of cross-organizational insurance claim processing [5].

Fig. 1 describes the workflows of each participant along with how they relate to one another. Notwithstanding that one could adopt a standard process notation, the main point is that such a description tightly couples the inner workings of the participants. Conventional protocol approaches [16, 25, 32] deemphasize the inner workings and capture the interactions between the participants via a formal notation in terms of constraints on the ordering of the messages exchanged between the participants.

In contrast, a commitment protocol emphasizes the social state of an interaction, expressed in terms of commitments. A commitment protocol describes the roles involved, the messages they exchange, and any preconditions 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.

3 Technical Approach

Positron provides a formal language in which to express composite protocols based on existing constituent protocols. Positron is a Java application that reads protocols described in the Positron language (examples shown in Listings 1 and 2), flattens any hierarchically nested protocols, and generates input to the MCMAS model checker.

Recall that Proton [14] provides a language for capturing roles, propositions, commitments, and messages. 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: Function *Req* for role requirements, *coupling commitments* for role accountabilities, and three *path expressions* for good and bad enactments.

Definition 1. *A* Positron protocol *is a six-tuple of $\mathcal{P}^{\mathsf{R}}$, a set of* role *names; $\mathcal{P}^{\mathsf{P}}$, a set of role-qualified propositions; $\mathcal{P}^{\mathsf{C}}$, a set of commitments; $\mathcal{P}^{\mathsf{M}}$, a set of guarded messages; $\mathcal{P}^{\mathsf{F}}$, a set of CTL expressions to be verified; and $\mathcal{P}^{\mathsf{U}}$, a set of use (include) statements.*

For brevity, we omit the Positron grammar in favor of examples. Listing 1 specifies AGFIL's Claim Handling protocol as roles (insured and claims handler), propositions, commitments, and messages (with their guards and effects on the propositions).

Listing 1 Claim constituent protocol

```
 1: protocol Claims (role In, role CH, prop notify, prop estimate, prop invoice, prop pay) {
 2:     role In; CH;
 3:     prop notify; estimate; invoice; pay;
 4:     commitment
 5:         cc = C_{In,CH}(estimate and invoice, pay);
 6:         notify.cc1 = C_{CH,In}(notify, estimate);
 7:         pay.c1 = C_{In,CH}(invoice, pay);
 8:         pay.c2 = C_{CH,In}(pay, invoice);
 9:     message
10:         In → CH : [true] notify_requestMsg means {notify};
11:         CH → In : [notify.isSet()] notify_responseMsg means {estimate};
12:         In → CH : [true] pay_do1 means {pay};
13:         CH → In : [true] pay_do2 means {invoice};
14: }
```

Listing 2 describes the AGFIL composite protocol. In the statement beginning on Line 4, this protocol uses the Claims protocol in Listing 1, mapping roles and propositions in protocol AGFIL to equivalent versions in protocol Claims. We omit the other constituent protocols for space. It turns out that the AGFIL protocol has no additional messages since all its messages derive from its constituents. The AGFIL protocol includes formulas describing the correctness requirements.

Listing 2 Positron specification for AGFIL protocol (partial)

```
 1: protocol AGFIL {
 2:     role PH; In; CC; CH; Re; Ad;
 3:     prop accident; deliver; repair; paid; ...
 4:     use IN-CH : Claims( agfil.Claims,
 5:         role In = IN, role CH=CH,
 6:         prop notify = true, prop estimate=reportCH,
 7:         prop pay = payCH, prop invoice=repairIn);
 8:       ...
 9:     commitment
10:       CC1 : C_{CC,{PH, Re}}(deliverReq, notifyRE);
11:       PH1 : C_{PH,{CC,Re}}(deliverReq ∧ approval, deliver);
12:       ...
13:     formula
14:       AG(paid ∧ accident → AF(repair ∨ anyCancel));
15:       Req(IN, coverage ∧ premium ∧ accident, repair);
16:       AG(¬(repair ∧ ¬inspectCH));
17:       ...
18:       EFPath(accident, deliverReq, payCa, reportCH, reportRe, ..., repair);
19:       ¬EFPath(repair, accident);
20:       ...
21: }
```

Definition 2. *An MCMAS representation is a tuple of $\mathcal{M}^{agent}$, a set of agent names, including a distinguished agent* Env *representing the environment; $\mathcal{M}^{state}$, a set of agent-qualified variable names; $\mathcal{M}^{msg}$, a set of agent-qualified guarded transition functions; $\mathcal{M}^{evol}$, a set of agent-qualified evolution expressions; $\mathcal{M}^{init}$, a set of agent-qualified variable initializations; $\mathcal{M}^{eval}$, a mapping from propositions to expressions over variables in $\mathcal{M}^{state}$; $\mathcal{M}^{fair}$, a set of fairness expressions; and $\mathcal{M}^{ctl}$, a set of CTL expressions to be verified.*

Protocol Composition. Positron supports nested composition of protocols. A composite protocol P can use (include) a parameterized constituent protocol with a use statement $q : Q(\overline{x} = \overline{p})$ specifying protocol name (q), protocol type (Q), a set of arguments $\overline{p}$ passed by a composite protocol P, and a matching set of 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 any hierarchical nesting in P to produce a single, flat 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 prepending the constituent name q to each element name in Q. Positron supports using multiple copies or instances of the same constituent Q by using distinct names q and q'.

Definition 3. *Given a set of arguments $\overline{p}$, a parameterized constituent protocol type Q accepts a set of parameters $\overline{x}$, where 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 $\overline{p}$.*

Expanding a composite P containing a constituent $q : Q(\overline{x} = \overline{p})$ yields the union of P and $Q^{\overline{x}}_{\overline{p}}$, and removing P's use statement $q : Q(\overline{x} = \overline{p})$.

Definition 4. *Given a composite protocol P that uses a constituent protocol $q : Q(\overline{x} = \overline{p})$, 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 $P' = \mathsf{expand}(P,\ q : Q(\overline{x} = \overline{p}))$ is the expanded version of P and Q, and is defined as follows, where $\mathsf{x} \in \{\mathsf{R}, \mathsf{P}, \mathsf{C}, \mathsf{M}, \mathsf{F}\}$*

$$\mathcal{P}^{\mathsf{x}}(P') := \mathcal{P}^{\mathsf{x}}(P) \cup \mathcal{P}^{\mathsf{x}}(Q^{\overline{x}}_{\overline{p}})$$
$$\mathcal{P}^{\mathsf{U}}(P') := (\mathcal{P}^{\mathsf{U}}(P) - q) \cup \mathcal{P}^{\mathsf{U}}(Q^{\overline{x}}_{\overline{p}}).$$

Positron Conversion to MCMAS. The conversion of a Positron protocol, $\mathcal{P}$, to an MCMAS representation $\mathcal{M} = \mathit{conv}(\mathcal{P})$ is a two-step process. First, constituent protocol expansion (Protocol Composition) flattens all nested protocols, ensuring $\mathcal{P}^{\mathsf{U}}$ is the empty set. Second, additional conversion functions, $\mathit{conv}^{p}_{m}(\mathcal{P}^{\mathsf{x}})$, convert each element of a Positron protocol to elements of an MCMAS representation. The final MCMAS representation consolidates these generated elements. The ISPL source input into MCMAS is generated from the above-mentioned MCMAS representation.

Role Requirements. A role requirement reflects a role-desired goal of an agent playing a role in the composite protocol. In Positron, $Req(r, p, q)$ means that role r requires that q will occur whenever p occurs. In AGFIL, one of POLICYHOLDER's role requirements is: if INSURER offers coverage, I paid the premium, and I have an accident, then my car will be repaired: Req(PH, *coverage* $\wedge$ *premium* $\wedge$ *accident*, *repair*)

It is incorrect to formalize a role requirement as the CTL specification: $\mathbf{AG}$(*accident* $\rightarrow$ $\mathbf{AF}$ *repair*), which ignores commitment violations that disrupt a collaboration: 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 express role r's requirement as: if r fulfills all its own commitments and p holds at any state, then on each branch eventually, either q holds or a role other than r canceled one of its commitments. If r's requirement fails because r cancels a commitment, that is not a fault of the protocol; it is r's fault.

A Positron role requirement maps to an MCMAS CTL expression as

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

where $r'.\mathit{anyCancel}$ is true if and only if role r' cancels any of its commitments.

Enactment Requirements Although capturing all possible enactments is not feasible for all protocols, designers and other stakeholders often know of specific good and bad enactments. We use these *enactments* for partially verifying a composite protocol as a way to assist designers refine protocol specifications (e.g., its constituent protocols and coupling commitments) or other requirements. An enactment corresponds to a scenario in requirements engineering [13] and yields a unit test. In Positron, the enactment specifications can be "good" (must exist) or "bad" (must not exist).

We use model checking to verify enactments. We introduce three recursive functions to simplify enactment specification. An enactment E is an ordered list of Boolean expressions over states and messages. *head*(E) is the first element in list E, and *tail*(E) is E without the first element, and $\mathbf{EX}$, $\mathbf{EF}$ and $\mathbf{EU}$ are CTL operators. Define

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

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

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

EXPath specifies a path of states that must appear consecutively. *EXPath* is often too strong a constraint, since it precludes interleaving of constituent protocols. *EFPath* specifies a path of states that must appear in order, but not necessarily consecutively. *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* requirement is stronger than *EFPath* and weaker than *EXPath*. Two example requirements from AGFIL include $\neg\mathit{EFPath}(\mathit{repair}, \mathit{accident})$ and $\mathit{EFPath}(\mathit{accident}, \mathit{deliverReq}, \mathit{deliverCar}, \ldots, \mathit{repair})$.

Coupling Commitments A composite protocol would in general relate its constituent protocols. All roles are jointly accountable for ensuring constituent protocols are properly interrelated. We capture each role's *role accountabilities* as *coupling commitments*. 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: $\mathsf{C}_{\text{accountable role},\{\text{interrelated roles}\}}(\mathit{antecedent}, \mathit{consequent})$.

Consider two coupling commitments from AGFIL. (1) CALLCENTER commits to POLICYHOLDER and REPAIRER that it will notify REPAIRER whenever it receives a request: $\mathsf{C}_{CC,\{PH,\,Re\}}(\mathit{deliverReq}, \mathit{notify}\text{RE})$. (2) POLICYHOLDER commits to CALLCENTER and REPAIRER that he will deliver his car to REPAIRER whenever requested and the repair request is approved: $\mathsf{C}_{PH,\{CC,Re\}}\ (\mathit{deliverReq} \wedge \mathit{approval}, \mathit{deliverCar})$.

Verification. Positron reads specifications of the composite and constituent protocols and generates a single MCMAS input file. MCMAS reads the input, builds the appropriate model, and reports whether each CTL formula holds in the model.

Fig. 2 shows a portion of the state space Positron generates for verification from AGFIL's constituent protocols and coupling commitments. The start state is s_0. Solid lines are valid transitions (messages); dashed lines are transitions that must not occur infinitely often. Since the message guard for *coverage* is *premium*, *coverage* can occur only after *premium*, making $s_1 \ldots s_8$ invalid start states. Notice that the top row

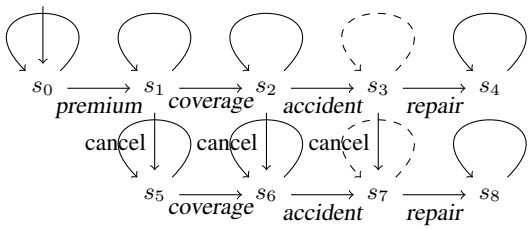

Fig. 2. Selected states and transitions for AGFIL.

(*premium*, *coverage*, *accident*, and *repair*) begins a good enactment. Positron can verify the existence of this path using an *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.

Positron generates a model checking fairness constraint for each commitment: a commitment must not remain unconditional and unresolved forever. Dashed loops are invalid because they violate a commitment fairness constraint. A composite protocol may fail to satisfy role or enactment requirements for different reasons:

- *Fail*$_{RR}$: If a role requirement (Req) formula fails, then coupling commitments are missing; add coupling commitments that require agents to act as appropriate.
- *Fail*$_G$: If a good enactment formula fails, then either (*Fail*$_{GG}$) some message guards are too strong; weaken guards to enable additional good transitions. Or (*Fail*$_{GC}$) some commitment can become detached, but can never resolve; weaken guards to enable transitions that satisfy the commitment's consequent.
- *Fail*$_B$: If a bad enactment formula fails, then some message guards are too weak; strengthen guards to disable existing incorrect transitions.

4 Evaluation of Positron Modeling and Tools

We evaluate our contributions by modeling real-life protocols from the insurance, manufacturing, and healthcare domains. Table 1 summarizes our iterative methodology to develop a composite protocol such as in Listing 2.

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

We create the AGFIL protocol of Listing 2 as follows. **Roles:** Identify roles: INSURER (IN) for agent AGFIL, CALLCENTER (CC) for EUROP ASSIST, CLAIMHANDLER (CH) for LEE, POLICYHOLDER (PH), REPAIRER (RE), and ADJUSTER (AD). **Constituent Selection:** Identify constituent protocols: *RequestResponse*, *Exchange* (where two roles swap items), *Claims* for IN-CH and *ApprovedWork* for CH-RE. **Role Requirements:** Identify role requirements: POLICYHOLDER requires: (1) if he has coverage, pays his premium, and has an accident, his car is repaired; (2) if he delivers

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

Step	Name	Inputs	Outputs
1	Roles	Background and requirements	Composite roles
2	Constituent Selection	Role relationships and protocol library	Constituent protocols
3	Role Requirements	Role's needs	Role requirements
4	Enactments	Background knowledge of requirements	Good and bad enactments
5	Coupling	Enactments	Coupling commitments Composite protocol
6	Positron	All artifacts	Protocol specification
7	Verification	Protocol specification	Model checker results

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. **Enactments:** Identify enactments: (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. **Coupling:** Identify coupling commitments: (1) Between messages (a) and (b) of the accident-reporting enactment (see previous step), if POLICYHOLDER reports an accident, CALLCENTER assigns and notifies REPAIRER and (2) between messages (c) and (d), if CALLCENTER asks POLICYHOLDER to deliver his car to REPAIRER, he does so. **Positron:** Generate the Positron specification for AGFIL. Listing 2 shows snippets of the Positron specification for AGFIL protocol. Lines 2 and 3 declare roles and propositions. Line 4 instantiates constituent *Claims* named IN-CH. Lines 10 and 11 are two coupling commitments. Line 15 lists one of POLICYHOLDER's role requirements. Line 16 lists an INSURER requirement as explicit CTL. Line 18 verifies the good, accident-reporting enactment that must exist in the composite, and Line 19 verifies a bad enactment that must not exist. Listing 1 is the Claims protocol used as a constituent protocol. From all previous artifacts, generate the MCMAS input files. **Verification:** Run MCMAS model checker.

Quote To Cash Evaluation. Quote To Cash (QTC) is an important business process that supports manufacturing supply chains [24]. **Roles:** Identify roles: CUSTOMER, RESELLER, DISTRIBUTOR, SELLER, SHIPPER1, and SHIPPER2. **Constituent Selection:** Identify constituent protocols: CUSTOMER orders goods and services from RESELLER using constituent protocol *CommercialTran* (*ComTran*), RESELLER fulfills the order by *Outsourcing* to DISTRIBUTOR, DISTRIBUTOR orders good from SELLER using *CommercialTran*, SELLER arranges shipping with SHIPPER2, and DISTRIBUTOR arranges shipping with SHIPPER1, using additional instances of *Outsourcing*, and SELLER provides a customer support contract to CUSTOMER though *StandingService*. **Role Requirements:** Identify role requirements; if CUSTOMER pays, he receives goods and services, if RESELLER pays DISTRIBUTOR, he receives shipment, and for CUSTOMER whenever a role performs its task, it gets paid. **Enactments:** Identify two enactments

for CUSTOMER placing an order and ending with fulfillment: one if DISTRIBUTOR has goods in stock, one if it restocks from SELLER. Fulfilling an order before it is verified is a bad enactment. **Coupling:** Identify coupling commitments: (1) CUSTOMER couples CU-RE and CU-RE-DI: if CUSTOMER receives a shipment, he pays RESELLER, (2) RESELLER couples CU-RE and CU-RE-DI: whenever RESELLER receives an order, he orders from DISTRIBUTOR. **Positron:** Generate Positron specification for QTC. **Verification:** Run MCMAS model checker.

Healthcare (ASPE) Evaluation. We consider the healthcare process for breast cancer diagnosis, as described by an HHS committee [2]. The resulting *ASPE* protocol contains five roles (for convenience, we associate feminine pronouns with PATIENT, RADIOLOGIST, and REGISTRAR and masculine pronouns with PHYSICIAN and PATHOLOGIST.) The process begins when PATIENT visits a primary care physician (PHYSICIAN), who detects a suspicious mass in her breast. He sends PATIENT to RADIOLOGIST for a mammography. If RADIOLOGIST notices suspicious calcifications, she sends a report to PHYSICIAN recommending a biopsy. PHYSICIAN requests the RADIOLOGIST to perform a biopsy, who collects a tissue specimen from PATIENT and sends it to a PATHOLOGIST. PATHOLOGIST analyzes the specimen, and performs ancillary studies. If necessary, PATHOLOGIST and RADIOLOGIST confer to reconcile their results and produce a consensus report. PHYSICIAN reviews the integrated report with PATIENT to create a treatment plan. PATHOLOGIST forwards his report to REGISTRAR who adds PATIENT to a state-wide cancer registry. There are only two coupling commitments in ASPE. PHYSICIAN has no coupling commitments because it is his choice whether PATIENT needs mammogram and biopsy exams from RADIOLOGIST.

Real-life Results: Positron and MCMAS were run on all three, hierarchical examples with the statistics and timings for the final, corrected protocols shown in Table 2. Positron processing was quick at less than 4 seconds, and total processing of AGFIL and ASPE were also quick at less than eight seconds. QTC, with a 1000 times larger state space and the most CTL formulas, required 21 minutes.

Table 2. Statistics. (M is 10^6, G is 10^9, s is seconds.)

Composite Metric	**AGFIL**	**QTC**	**ASPE**
Constituent instances	11	6	12
Roles	6	6	5
Propositions	22	37	18
Commitments (total)	24	43	12
Coupling commitments	9	21	2
Messages	22	55	20
CTL Formulas (total)	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

Model Verification: Resolving verification errors is a challenging task, requiring careful consideration of the interactions between the generated MCMAS model and CTL statements.

Positron helped identify and fix several verification failures. Good enactment failures ($Fail_{GG}$ and $Fail_{GC}$) were generally easier to fix, since they only require that some path exist. One or more requirements or coupling commitments were slightly weakened

to allow additional branches at appropriate points. Bad enactment failures ($Fail_B$) were harder to fix as they required the impossibility of a particular enactment. One or more role requirements or coupling commitments were added. Hand checking was insufficient to ensure the changes would eliminate all bad paths; only reruns of Positron and MCMAS could confidently verify the changes.

Model Validation: Although identifying requirements was easy, some initial specifications were incorrect because preconditions were missed. For example, POLICYHOLDER's role requirement initially failed ($Fail_{RR}$) because coupling commitment among CALLCENTER, POLICYHOLDER, and REPAIRER did not include *approval*; REPAIRER will not 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, one of the AGFIL's good enactment 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.

5 Discussion: Literature and Future Work

Positron gains an advantage over both traditional process modeling and existing (operational) protocol approaches by focusing on high-level 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 failing role.

Composite protocols provide a formal means to capture how constituent protocols may be composed to realize an STS specification. Because these protocols capture meanings as commitments (generalizable to norms), yet have a formal semantics that maps to sound enactments, they can provide a natural approach to support secure policy-governed collaboration in ways that are not visible to low-level, operational approaches.

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 sociotechnical 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. [9] propose OWL-P [10] and MAD-P [11] for specifying and verifying commitment protocols and their compositions. They employ axioms to specify a composition. These approaches suffer from a key drawback: axiom violations are not assigned to any particular role. In contrast, Positron employs coupling commitments with clear role accountability for the effects of one constituent protocol on others. Further, Amoeba is purely manual, whereas Positron incorporates mechanical verification.

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., B=Baldoni et al.. Check marks show the significant topics addressed by each paper. The cell contents of the verification rows indicate whether the paper discusses (D) or mechanizes (M) verification of known good or bad paths.

Significant Topics / **Verification Topics**	**Po**	**Pr** [14]	**DA** [9]	**DO** [10]	**Dv** [8]	**DM** [11]	**T** [31]	**Y** [33]	**Mi** [21]	**G** [15]	**C** [6]	**Mc** [20]	**L** [18]	**B** [4]
Protocol specification	✓	✓	✓	✓	–	✓	✓	✓	✓	✓	✓	✓	–	✓
Methodology	✓	–	✓	✓	–	–	✓	–	–	✓	✓	–	–	–
Single protocol	–	✓	✓	✓	✓	–	–	✓	✓	✓	–	✓	✓	✓
Protocol patterns	–	–	–	–	–	–	✓	–	–	–	–	–	–	–
Protocol-to-protocol verification	–	✓	–	–	–	–	?	–	–	–	–	?	?	–
Protocol composition	✓	–	✓	✓	✓	✓	–	–	✓	–	✓	–	✓	✓
Requirements verification														
Sociotechnical	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

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 [31] (T&S) describe a methodology for modeling STSs that captures the commitments to be created among the parties by melding selected collaboration 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 [33] 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 STS requirements. It would be interesting to formulate Yolum's generic correctness properties in Positron.

Miller and McBurney [21] (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. Theses are important in naturally describing STS protocols, as we demonstrated above. Whereas M&M describe a custom reasoner, we rely on standard CTL semantics as realized in MCMAS.

Günay et al. [15] 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 [6] describe the Hermes system for goal-oriented interaction. They focus on interaction-level goals, whereas we focus on role-level requirements and commitments. Their action sequence diagrams capture only good enactments.

McGinnis and Robertson [20] 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 agents from making an undesirable change to a protocol. If their protocols were augmented with commitments, Positron could 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. While they describe rules for dynamically changing protocols, they do not address formal verification of interaction properties.

Lomuscio et al. [18] semiautomatically compile and verify contract-regulated service compositions with compliance expressed in temporal-epistemic logic using MCMAS (which we adopt). A crucial difference is that Lomuscio et al. consider service compositions; we consider protocol compositions. Since a protocol has a distributed footprint, protocol compositions are inherently more subtle than service compositions. A potential benefit from adopting MCMAS is that it supports more expressive logics such as Alternating-Time Temporal Logic (ATL) [1], which could help capture subtle correctness criteria for protocols.

We propose a methodology and use role responsibilities and role accountabilities using expressions in CTL. Others describe the complementary issue of *temporal pattern languages* which assist users to correctly capture their high level requirements as temporal expressions. Dwyer et al. [12] describe a pattern language for temporal expressions in CTL, LTL and other formalisms. Baldoni et al. [4] compose protocols using regulative specifications as LTL constraints.

BPMN 2.0 [23] is a standard notation for business process modeling. BPMN addresses both orchestration and choreography; in taking a multiagent approach, Positron focuses on choreography. Our protocols are similar to BPMN Conversation objects. Both protocols and Conversations can involve two or more roles (Participants), and both can have simple or complex message sequencing relationships. But, BPMN does not specify how to describe the relationships between messages in a complex Conversation, except through internal orchestration; our complex protocols fully specify message sequencing relationships using external guard statements. Both support arbitrary nesting. We formally verify our compositions using temporal logic; model verification is explicitly out-of-scope in the BPMN specification. Positron's primary building blocks are protocols and commitments. BPMN has no counterpart to our coupling commitments, which are key to our interrelating constituent protocols and formal verification.

Threats, Limitations, and Future Directions. One limitation of Positron is that it does not handle varieties of accountability besides commitments [29] and does not

show how to evaluate agent goals and decision making with respect to protocols [15]. Importantly, we have not established that practitioners employing Positron can obtain the benefits in abstraction, reusability, and correctness the motivate it.

These threats and limitations lead to useful future directions. At the theoretical level, treating the goals of the participants 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 collaboration in STSs would be necessary to promote adoption by industry. To this end, we have developed a graphical notation for protocol composition called *composite protocol diagrams* (CPDs). CPDs seek to succinctly visualize the essence of a composite protocol both to analysts and technical designers, who collaborate in its construction. We defer an empirical evaluation of CPDs and competing notations as a way to determine if the high-level abstractions of Positron can help analysts and designers combat complexity and communicate more effectively with each other.

Acknowledgments

Thanks to the anonymous reviewers for helpful comments and to the US Department of Defense for partial support through a Science of Security Lablet grant.

References

1. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. J. of the ACM 49(5), 672–713 (Sep 2002)
2. ASPE: The importance of radiology and pathology communication in the diagnosis and staging of cancer (Nov 2010), Assistant Secretary for Planning and Evaluation, U.S. Department of Health and Human Services; http://aspe.hhs.gov/sp/reports/2010/PathRad/index.shtml
3. Baldoni, M., Baroglio, C., Marengo, E.: Behavior-oriented commitment-based protocols. In: Proc. 19th European Conf. on Artif. Intell. (ECAI). pp. 137–142 (Aug 2010)
4. Baldoni, M., Baroglio, C., Marengo, E., Patti, V., Capuzzimati, F.: Engineering commitment-based business protocols with the 2CL methodology. J. Auton. Agents Multi-Agent Syst. (JAAMAS) 28(4), 519–557 (Jul 2014)
5. Browne, S., Kellett, M.: Insurance (motor damage claims) scenario. Document D1.a, CrossFlow Consortium (1999)
6. Cheong, C., Winikoff, M.P.: Hermes: Designing flexible and robust agent interactions. In: Dignum, V. (ed.) Handbook of Research on Multi-Agent Systems, chap. 5. IGI Global (2009)
7. Chopra, A.K., Dalpiaz, F., Aydemir, F.B., Giorgini, P., Mylopoulos, J., Singh, M.P.: Protos: Foundations for engineering innovative sociotechnical systems. In: Proc. 18th IEEE Intl. Requirements Engg. Conf. (RE). pp. 53–62. (Aug 2014)
8. Desai, N., Cheng, Z., Chopra, A.K., Singh, M.P.: Toward verification of commitment protocols and their compositions. In: Proc. 6th Intl. Joint Conf. Auton. Agents Multiagent Systems. pp. 33:1–33:3. ACM (2007)
9. Desai, N., Chopra, A.K., Singh, M.P.: Amoeba: A methodology for modeling and evolving cross-organizational business processes. ACM Trans. Soft. Engg. Methodology (TOSEM) 19(2), 6:1–6:45 (Oct 2009)
10. Desai, N., Mallya, A.U., Chopra, A.K., Singh, M.P.: Interaction protocols as design abstractions for business processes. IEEE Trans. Soft. Engg. 31(12), 1015–1027 (Dec 2005)

11. Desai, N., Singh, M.P.: A modular action description language for protocol composition. In: Proc. 22nd Conf. Artif. Intell. (AAAI). pp. 962–967. (Jul 2007)
12. Dwyer, M.B., Avrunin, G.S., Corbett, J.C.: Patterns in property specifications for finite-state verification. In: Proc. Intl. Conf. Soft. Eng. (ICSE). pp. 411–420. (May 1999)
13. Filippidou, D.: Designing with scenarios: A critical review of current research and practice. Requirements Engg. 2(1), 1–22 (Mar 1998)
14. Gerard, S.N., Singh, M.P.: Formalizing and verifying protocol refinements. ACM Trans. Intelligent Systems and Technology (TIST) 4(2), 21:1–21:27 (Mar 2013)
15. Günay, A., Winikoff, M., Yolum, P.: Commitment protocol generation. In: Proc. 10th AAMAS Wkshp. Decl. Agent Lang. Tech. (DALT). pp. 51–66 (Jun 2012)
16. HL7: Health Level Seven (2007), http://www.hl7.org
17. Lomuscio, A., Qu, H., Raimondi, F.: MCMAS: A model checker for the verification of multi-agent systems. In: Proc. Intl. Conf. Comput. Aided Verif. pp. 682–688. LNCS (2009)
18. Lomuscio, A., Qu, H., Solanki, M.: Towards verifying contract regulated service composition. J. Auton. Agents Multi-Agent Syst. (JAAMAS) 24(3), 345–373 (May 2012)
19. Mallya, A.U., Singh, M.P.: An algebra for commitment protocols. J. of Autonomous Agents and Multi-Agent Systems (JAAMAS) 14(2), 143–163 (Apr 2007)
20. McGinnis, J., Robertson, D.: Dynamic and distributed interaction protocols. In: Proc. Wkshp. Adaptive Agents and Multi-Agent Systems. LNCS 3394, pp. 167–184. Springer (2005)
21. Miller, T., McBurney, P.: Propositional dynamic logic for reasoning about first-class agent interaction protocols. Computational Intelligence 27(3), 422–457 (2011)
22. Miller, T., McGinnis, J.: Amongst first-class protocols. In: Proc. 8th Intl. Wkshp. Engg. Soc. Agents World (ESAW 2007). LNCS 4995, pp. 208–223. Springer (2008)
23. Object Management Group: Business Process Model and Notation (2011). http://bpmn.org/
24. Oracle: Automating the Quote-to-Cash process (Jun 2009), http://www.oracle.com/us/industries/045546.pdf
25. RosettaNet: Home page (2009), http://www.rosettanet.org
26. Singh, M.P.: An ontology for commitments in multiagent systems: Toward a unification of normative concepts. Artificial Intelligence and Law 7(1), 97–113 (Mar 1999)
27. Singh, M.P.: Information-driven interaction-oriented programming. In: Proc. 10th Intl. Conf. Auton. Agents MultiAgent Syst. (AAMAS). pp. 491–498 (May 2011)
28. Singh, M.P.: Commitments in multiagent systems. In: Paglieri, F. et al., (eds.) The Goals of Cognition, chap. 32, pp. 613–638. College Publications (2012)
29. Singh, M.P.: Norms as a basis for governing sociotechnical systems. ACM Trans. Intelligent Systems and Technology (TIST) 5(1), 21:1–21:23 (Dec 2013)
30. Telang, P.R., Singh, M.P.: Abstracting and applying business modeling patterns from RosettaNet. In: Proc. 8th ICSOC. LNCS 6470, pp. 426–440. Springer (2010)
31. Telang, P.R., Singh, M.P.: Specifying and verifying cross-organizational business models: An agent-oriented approach. IEEE Trans. Services Computing 5(3), 305–318 (Jul 2012)
32. WS-CDL: Web Services Choreography Description Language, version 1.0 (Nov 2005), http://www.w3.org/TR/ws-cdl-10/
33. Yolum, P.: Design time analysis of multiagent protocols. Data and Knowledge Engineering J. 63(1), 137–154 (Oct 2007)
34. Yolum, P., Singh, M.P.: Commitment machines. In: Proc. 8th Intl. Wkshp. Agent Theories, Architectures, and Languages (ATAL 2001). LNAI 2333, pp. 235–247. Springer (2002)