System Safety as an Emergent Property in Composite Systems

Jennifer Black
Carnegie Mellon University
Pittsburgh, PA USA
jenmorris@cmu.edu

Philip Koopman
Carnegie Mellon University
Pittsburgh, PA USA
koopman@cmu.edu

Abstract

Decomposition is used to manage system complexity, but is problematic for emergent properties such as system safety. Previously, we introduced Indirect Control Path Analysis (ICPA) for elaborating system safety goals in composite systems. We now provide mathematical definitions of emergent and composable system behaviors in the context of formal specifications and ICPA, and identify useful special cases in which partial decomposition of emergent safety goals is possible. We apply ICPA to a semi-autonomous automotive system to identify safety goals for key subsystems, and then monitor the system and subsystem goals at run-time in an implementation of the vehicle. Although false negatives at the subsystem level indicate the subgoals do not fully compose the original safety goal, some system-level goal violations are detected by subsystem monitors. In addition, monitoring at both the system and subsystem level has identified certain safety-related errors that may be imperceptible to system testers.

1. Introduction

In distributed embedded systems, a common strategy for managing complexity is to divide the system into subsystems by functionality. Then each subsystem is designed and developed separately, often by different groups within a company or by outside vendors. In a distributed automotive system, for example, different controllers may be allocated to handle engine, transmission, and stability algorithms, each with a separate set of requirements that describes its functional behaviors.

Unfortunately, decomposing non-functional requirements, also known as goals [8] or quality attributes [12][2], is not straightforward. Quantifiable goals, such as cost or performance, may be decomposed by allocating a fixed limit on each component [22]. However, system safety, defined by Leveson as "freedom from accidents or losses" [17], is usually not expressible as a sum of parts. Consider an automotive safety goal: "the vehicle shall experience zero collisions." Unlike performance goals, where the concept of "time" is the same for components and system, the concept of "collision" at the system level does not have the same meaning for individual components. These properties are said to be *emergent* [20].

In [5], we introduced Indirect Control Path Analysis (ICPA) for elaborating system safety goals across a composite system. Safety goals were defined for a semi-autonomous automotive system from a commercial automotive research lab with five active safety/driver convenience features: Collision Avoidance (CA), Rear Collision Avoidance (RCA), Adaptive Cruise Control (ACC), Lane Change Assist (LCA), and Park Assist (PA). ICPA was then applied to identify safety goals for each feature subsystem and the subsystem that arbitrates among features. Other vehicle subsystems were excluded from the analysis, but the technique could be applied to them as well.

This initial work provided a set of subgoals, but it was unknown whether or not these subgoals were sufficient to satisfy the original parent goal. If a vehicle's safety goals are indeed emergent, it may not be possible to identify all subgoals that meet the parent goal. However, the set of safety subgoals produced with ICPA may still be useful if it is possible to partition an emergent safety goal into two parts: one part that is composable and another that is emergent.

In this paper, we define emergent and composable behaviors in the context of formally specified goals and ICPA, and identify useful special cases in which emergent system goals may be partially composable. To evaluate the technique, run-time monitors for the semi-autonomous automotive system safety goals and subgoals were added to an implementation of the vehicle in a simulation environment. False negative detection at the subsystem level in some of the scenarios suggests the set of subsystem safety goals do not fully satisfy the system-level behavior. In such cases, the system safety goal may be a truly emergent property, detectable

Preprint from 2009 International Conference on Dependable Systems and Networks (DSN09)

P	true in current state
$\neg$P	false in current state
●P	true in previous state
♦P	true in some previous state
■P	true in all previous states
○P	true in next state
◇P	true in current or some future state
□P	true in current and all future states
P $\rightarrow$ Q	P implies Q in current state
P $\Rightarrow$ Q	□(P $\rightarrow$ Q); P implies Q in all states
P $\Leftrightarrow$ Q	P *iff* Q in all states
$\bullet\blacksquare_{<T}$P	true for duration T in previous state
$\bullet\blacklozenge_{<T}$P	true at least once in duration T in previous state
@P	●$\neg$P $\wedge$ P; true in current state, false in previous

Figure 1. Temporal logic operators

only by a system-level monitor. However, the subsystem monitors did detect some situations in which the parent goal was violated, and monitors at both the system and subsystem levels detected violations that may be imperceptible to system testers driving the vehicle. In addition, suspected design defects identified during the ICPA process were confirmed at run-time.

2. Background and previous work

The focus of this research is elaborating emergent system safety goals across a composite system. This section reviews formal specification of composite systems, prior work on the ICPA technique, and emergence.

2.1. Formal specification of composite systems

Formal reasoning about specification of composite systems was introduced in [11]. Component behavior specifications were derived by pruning (eliminating behaviors which violate a constraint) and decomposing constraints to assign to agents. Mylopoulos et al. [22] presented a process-oriented approach for specifying non-functional requirements that defined them in terms of goals, links between goals, methods for goal refinement, correlation rules, and a labeling structure for linking goals to design decisions. Intent specifications [18] and safety patterns [3][4] are two additional approaches to specifying non-functional requirements that provide frameworks for expressing system safety goals, but do not offer tactics for safety goal decomposition.

The approach proposed in this paper builds primarily upon Goal-Oriented Requirements Engineering (GORE) as defined by the KAOS framework [8]. In goal-oriented requirements elicitation, goals are constructed in temporal logic expressions [13]. Figure 1 lists the temporal operators of KAOS used in this paper. In the KAOS framework, *goal realizability* is driven by monitorability and controllability of system state variables [15]. A goal relation can be expressed as $G_{(M,C)}$, where G is the goal to be realized in the system, M is the set of variables in the goal to be monitored, and C is the set of variables in the goal to be controlled. A goal is realizable by an agent if M is a subset of the variables monitored by agent ag and C is a subset of the variables controlled by agent ag, ($M \subseteq Mon(ag)$ and $C \subseteq Ctrl(ag)$). A goal may also be realizable by a set of realizable subgoals that satisfy the parent goal without being more restrictive.

One advantage of using formal specifications is that goal structure can guide elaboration. High-level goals are refined into sub-goals by *AND/OR* reductions [23], by applying formal refinement patterns for logical expressions [9], and by defining subgoals that are realizable by particular agents [15]. Goal patterns also guide operationalization into constraints and triggers for actions performed by agents in the system [16]. In addition, there are several techniques for identifying and resolving conflicts between goals [14].

2.2. Indirect Control Path Analysis (ICPA)

The aim of goal elaboration is to define subgoals that meet the parent goal and are realizable by agents in the system (i.e. can be operationalized) [15]. An additional aim of *safety* goal elaboration is to employ a goal coverage strategy. Additional tactics are needed for safety because existing tactics define subgoals that exactly meet the parent goal without being more restrictive or redundant [15]. Safety goals may require redundancy or restriction beyond exactly meeting the parent goal in order to take into account component failures.

Indirect Control Path Analysis (ICPA) [5] was introduced as a way to identify subgoals for subsystems from a parent safety goal. ICPA is a top-down search, similar to Fault Tree Analysis (FTA) [17]. Whereas FTA traces a top-level hazard to its lower-level causal events, ICPA traces a top-level state variable through the design to the components that influence it. ICPA uses a table structure similar to Failure Modes and Effects Analysis (FMEA) [17] to record these indirect control relationships. Agents along the trace path belong to the indirect control path and require further analysis of their relationships to the root variable. Leveson's Systems-Theoretic accident model and Process hazard Analysis (STPA) uses a similar approach based on the control architecture of the system [19]. Unlike ICPA, which uses backward search from safety goals defined at the system level through the control architecture to define safety goals for components, STPA uses forward search from component outputs to system hazards.

A *goal coverage strategy* is a plan for allocating subgoals to ensure that a high-level goal is met. Each strategy is defined by *goal assignment* and *goal scope*. Goal assignment defines which indirect control sources have subgoals and how those subgoals relate to each other (e.g., single responsibility, redundant responsibility, and coordinated responsibility). Goal assignment may be driven by physical limitations of the system (e.g., actuation delays) or by possible loss of monitorability and controllability by agents in the system. Goal scope defines how closely the safety subgoals meet the system safety goal. Although it may be possible for agents to meet the original safety goal exactly, it may sometimes be necessary or desirable to assign subgoals that are more restrictive than the original safety goal. Detailed information about how to apply ICPA can be found in [5].

2.3. Emergent properties

Goal elaboration is complicated by the emergent nature of system safety. Emergence had been observed in natural systems as early as the time of Aristotle, who recognized there were systems in which, "the whole is something beside the parts" [1]. The term *emergent* was first defined by Lewes [20], who asserted that a *resultant* system could be expressed as a sum or product of its components' outputs, whereas the outputs of components in an *emergent* system are fundamentally different from each other and from the resulting system behavior:

> Add heat to heat and there is a measurable resultant; but add heat to different substances, and you get various effects, qualitatively unlike...

Emergence has also been studied in complex systems such as artificial life [7][10] and agent-based systems [25]. Cariani [7] differentiates between three types of emergence: *computational*, *thermodynamic*, or *relative to a model*. Whereas computational emergence focuses on identifying emergent behaviors that arise from local computations, emergence relative to a model is more concerned with how emergent properties of the system affect the behavior of its components.

Darley [10] asserts that, similar to the halting problem [26], determining whether or not a system is emergent is undecidable, and the best predictor of system behavior is simulation. It may be possible to observe emergent behavior (something that is not predicted by individual component behaviors) as the system runs, but it is impossible to know for sure that a system is non-emergent, or even that all emergent behaviors in the system have been exposed.

Privosnik [25] proposed defining emergence at various levels of the hierarchy in agent-based systems. Component behaviors contribute to emergent behaviors at the subsystem level, which contribute to new emergent behaviors at the system level. Thus, system behavior is achieved by defining the behaviors of combinations of agents, as well as the agents themselves.

3. Emergence and goal specification

In general, emergence is problematic because its behaviors are unexpected. In telecommunications this is called the feature interaction problem, in which adding features to a system produces new behaviors not included in any single feature [6]. In system safety, emergence causes two types of problems. The first occurs when interaction among components produces hazardous behaviors that are previously unidentified (i.e., some system safety goals are missing or incorrect). If the set of system safety goals is incomplete, then the set of subgoals obtained by goal elaboration will fail to achieve system safety. The second problem occurs when interaction among components produces emergent behaviors that violate a known system safety goal (i.e., the system was designed and built to satisfy the goal but ultimately did not). In this case, the parent goal is not fully satisfied by the set of subgoals.

In order to understand and handle emergence in system safety, emergence must be defined within the context of the formal specification of the safety goals.

3.1. Emergent and composable behaviors

In GORE and ICPA, a goal is a temporal logic expression of state variables, or some combination of expressions by logical conjunction (connected by an *AND* operator) or disjunction (connected by an *OR* operator).

We define goal G to be *fully composable* if there exists a set of n subgoals $\{G_1, G_2, G_3, ...G_n\}$ that are realizable by one or more components, such that G is materially equivalent to the conjunction of the subgoals:

$$G_1 \wedge G_2 \wedge G_3 ... \wedge G_n \Leftrightarrow G \qquad (1)$$

which is equivalent to the conjunction of expressions:

$$G_1 \wedge G_2 \wedge G_3 ... \wedge G_n \Rightarrow G \qquad (2)$$

$$\neg G_1 \vee \neg G_2 \vee \neg G_3 ... \vee \neg G_n \Rightarrow \neg G \qquad (3)$$

Thus, the parent goal is satisfied (evaluates to *TRUE*) when all subgoals are satisfied and is violated (evaluates to *FALSE*) when any subgoal is violated.

We define goal G to be *emergent* if there is no set of subgoals realizable by any component in the system that satisfy both Equation (2) and Equation (3). We define goal G to be *emergent but partially composable* if there exists a set of m subgoals $\{G_1, G_2, G_3, ...G_m\}$ that are realizable by one or more components in the system, plus some undefined or unrealizable goal(s) X, such that G is materially equivalent to the conjunction of the realizable and undefined subgoals:

$$G_1 \wedge G_2 \wedge G_3 ... \wedge G_m \wedge X \Leftrightarrow G \qquad (4)$$

which is equivalent to the conjunction of expressions:

$$G_1 \wedge G_2 \wedge G_3 ... \wedge G_m \wedge X \Rightarrow G \qquad (5)$$

$$\neg G_1 \vee \neg G_2 \vee \neg G_3 ... \vee \neg G_m \vee \neg X \Rightarrow \neg G \qquad (6)$$

Composability of a goal is the extent to which X can be minimized.

Subgoals for a partially composable system goal are not very useful when the objective of the goal is to *produce* a behavior. In order to produce behavior A in Equation (5), all of the subgoals G_1 through G_m and the unknown goal(s) X would have to be satisfied. Without knowing X it is impossible to say anything about whether or not G is satisfied.

Alternatively, if the real objective is to *prohibit* a behavior, then a partially composable system goal may be of some use. Safety goals specify the "safe" state of the system to be maintained, but the real objective of system safety is to prevent a hazardous state from occurring. If goal G is some safety goal, then $\neg G_m$ represents a violation of the goal that puts the system in a hazardous state. In this situation it is useful to know some specific conditions in which the goal is violated, even if it is impossible to know all conditions that satisfy the goal. In Equation (6), we see that the goal is violated if any of goals G_1 through G_n or the unknown/unrealizable goal(s) X are violated. Although it is impossible to guarantee the system will always be safe without knowing X, it is possible to eliminate some known conditions that put the system into a hazardous state.

In other words, it may be possible to specify some, but not all component behaviors or interactions that cause a hazard. For example, it may be known that a vehicle will accelerate if a subsystem applies the throttle. However, there may be additional unknown interactions in the vehicle that also cause vehicle to accelerate (e.g., loss of brake fluid combined with a slick, sloped road). If a safety goal restricts accelerations over a certain threshold, it is important to prevent known sources of acceleration from violating the goal, in this case by limiting the acceleration caused by a subsystem.

These definitions apply to compositions without redundant goal coverage. When redundant subgoal sets are used, the equations inlcude the redundant sets connected by OR operators.

3.2. Emergence and ICPA

In all but the most simple systems, some degree of emergence is unavoidable. There will always be a component whose behavior cannot be fully specified or an unanticipated interaction among components. X in Equation (4) could exist because you don't know what to look for. Other times, X might occur if observability and controllability requirements of the goal are not met by any subsystem or combination of subsystems.

The purpose of ICPA is to identify subgoals for subsystems that satisfy the system-level safety goal. How close the set of subgoals come to satisfying the parent goal depends on how the indirect control paths are defined. The set of subgoals generated by ICPA will not satisfy the parent goal if sources of indirect control are omitted from the analysis, or relationships between these sources are incorrectly defined. However, just as it is impossible to determine whether a system is not emergent, it is also impossible to know if maximum composability has been extracted from an emergent but partially composable goal. In other words, it is impossible to determine if X is minimized in Equation (4).

In general, if a goal can be expressed as a conjunction of expressions, then it may be partially composable. A goal of the form:

$$\Box(A \wedge X) \tag{7}$$

can be divided into two subgoals:

$$\Box A \tag{8}$$

$$\Box X \tag{9}$$

Another goal of the form:

$$A \vee X \Rightarrow B \tag{10}$$

which is equivalent to:

$$\Box((\neg A \wedge \neg X) \vee B) \tag{11}$$

can be divided into two subgoals:

$$A \Rightarrow B \tag{12}$$

$$X \Rightarrow B \tag{13}$$

If X is unknown or unrealizable, the behaviors required by goals (8) and (12) may still be ensured, even if all behaviors required by goals (7) and (10) cannot. Consider a goal that requires a vehicle to be stopped whenever there is an object in the vehicle path. If object detection is non-ideal, then the goal cannot be fully realized. This goal, including the uncertainty, can be expressed in the same form as Equation (10):

$$\begin{aligned} &InPathDetected \ \vee \ InPathNotDetected \\ &\qquad\qquad\qquad \Rightarrow \ StopVehicle \end{aligned} \tag{14}$$

An emergent goal with unknown or unrealizable subgoals may still be satisfied by making it more restrictive in one of two ways. First, an *OR* reduction may be applied to disjunctions [23]. A goal of the form:

$$\Box(A \vee X) \tag{15}$$

where X is an unknown or unrealizable subgoal, can be satisfied by the more restrictive goal:

$$\Box A \tag{16}$$

Likewise, a goal of the form:

$$A \wedge X \Rightarrow B \tag{17}$$

which is equivalent to:

$$\Box(\neg A \vee \neg X \vee B) \tag{18}$$

can be satisfied by the more restrictive goal:

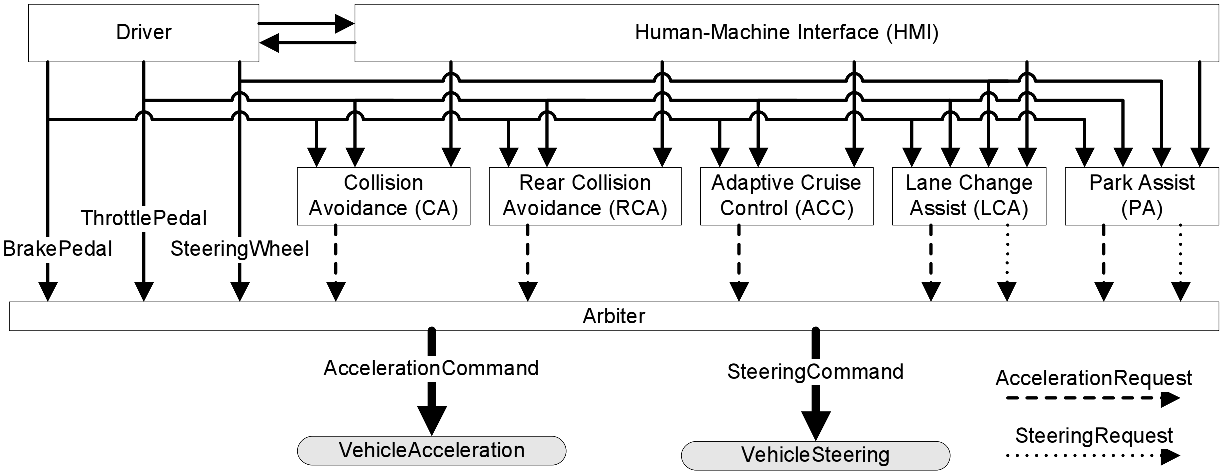

Figure 2. Semi-autonomous automotive system

$$A \Rightarrow B \tag{19}$$

OR reduction is more restrictive because some functionality of the system that would be otherwise be acceptable is prohibited. In equation 17, the state $A \wedge \neg X \wedge \neg B$ is acceptable, but in equation 19 it is not.

Another way to make the goal more restrictive is to increase the safety envelope of a given variable. Consider a goal that prohibits vehicle acceleration above a certain threshold $AccelerationLimit$. A subgoal for subsystems that control vehicle acceleration might prohibit vehicle acceleration above a lower threshold, perhaps $AccelerationLimit - SafetyEnvelope$.

At run-time, if the system goals and subgoals are not materially equivalent (i.e., the goal is not fully composed by the subgoals), false positive and false negative detections may result. A violation of a subgoal that does not correspond to a violation of its parent goal is considered a false positive (the subgoals indicate a hazardous state but the system is not in a hazardous state). A violation of a goal that does not correspond to a violation of a subgoal is considered a false negative (the subgoals do not indicate a hazardous state but the system is in a hazardous state). False negatives can occur when the unknown/unrealizable subgoal(s) are the cause of the goal violation. False positives can occur when the subgoals are more restrictive than the original safety goal.

4. Semi-autonomous automotive system

ICPA was applied to an automotive vehicle with safety-critical semi-autonomous features from an automotive research lab. The vehicle was intended as a platform for research in sensor fusion and autonomous control, and was not a production system. At the time of this work, the system design, implementation, and simulation environment were functional but incomplete. The vehicle subsystems included in this study are shown in Figure 2.

The feature subsystems include two active safety features and three driver convenience features. Collision Avoidance (CA) detects objects in the forward path and stops the vehicle before a collision occurs. Rear Collision Avoidance (RCA) performs a similar function when the vehicle is moving in reverse. Adaptive Cruise Control (ACC) commands the vehicle to a speed set by the driver, or to a set follow distance behind a slower lead vehicle. Lane Change Assist (LCA) works in conjunction with ACC to perform a lane change maneuver when requested by the driver. Park Assist (PA) finds a parking space and parks the vehicle, also when requested by the driver.

This automotive system is semi-autonomous because subsystems control vehicle motion under driver supervision. In CA and RCA, the driver has primary responsibility for detecting imminent collisions and avoiding them by some means, such as slowing, stopping, or steering the vehicle. In this case, CA and RCA provide backup safety behavior if the driver fails to stop the vehicle in time to avoid the collision. In ACC, LCA, and PA, the driver is responsible for monitoring the feature behaviors and intervening if needed (e.g. if a deer is heading toward the road but not yet detectable in the vehicle path). The driver may override any feature behavior except an emergency stop, in which case the driver is only permitted to brake harder than the feature.

4.1. Vehicle-level safety goals

Requirements for the research vehicle were not written in a formal specification language. In addition, there were no specific safety goals defined in the requirements documentation. Thus, our first step was to identify and formally specify the system safety goals. These safety goals were derived from inspection of the functional requirements, the hazard analysis that had been performed on the preliminary system requirements, and

Table 1. Safety goals for semi-autonomous automotive system

System Safety Goals	
S-01	**Goal:** Achieve[AutoAccelBelowThreshold] **InformalDef:** *Vehicle acceleration caused by autonomous vehicle control shall not exceed* $2\ m/s^2$. **FormalDef:** $\forall$ va: VehicleAcceleration; IsSubsystem(va.source) $\Rightarrow$ va.value $\leq 2\ m/s^2$
S-02	**Goal:** Achieve[AutoJerkBelowThreshold] **InformalDef:** *Vehicle jerk caused by autonomous vehicle control shall not exceed* $2.5\ m/s^3$. **FormalDef:** $\forall$ vj: VehicleJerk; IsSubsystem(vj.source) $\Rightarrow$ vj.value $\leq 2.5\ m/s^3$
S-03	**Goal:** Achieve[SubsystemAccelSteeringAgreement] **InformalDef:** *If a subsystem requests control of acceleration and steering and is granted control of either acceleration or steering, the subsystem shall control both acceleration and steering.* **FormalDef:** $\forall$ ac: AccelerationCommand, sc: SteeringCommand, sn: SubsystemName $\bullet$RequestingAcceleration(sn) $\wedge$ $\bullet$ RequestingSteering(sn) $\wedge$ ((ac.source = sn) $\vee$ (sc.source = sn)) $\Rightarrow$ (ac.source = sc.source = sn) $\wedge$ (ac.value = $\bullet$RequestedAcceleration(sn)) $\wedge$ (sc.value = $\bullet$RequestedSteering(sn))
S-04	**Goal:** Achieve[NoAutoAccelFromStop] **InformalDef:** *If a) the vehicle is stopped for a duration of* StoppedTime *and b) the throttle pedal has not been applied within the preceding* AccelerationDelay *and c) a subsystem is controlling acceleration and d) the HMI has not sent a* go *signal to the controlling subsystem within the preceding* AccelerationDelay*, then there shall be no vehicle acceleration.* **FormalDef:** $\forall$ va: VehicleAcceleration, vsp: VehicleSpeed, sn:SubsystemName, hmi: HumanMachineInterface, tp: ThrottlePedal, st: StoppedTime, ad: AccelerationDelay $\bullet\blacksquare_{<st}$IsStopped(vsp.value) $\wedge \neg\bullet\blacklozenge_{<ad}$ Applied(tp) $\wedge$ (va.source = sn) $\wedge \neg\bullet\blacklozenge_{<ad}$ Applied(hmi.go, sn) $\Rightarrow$ NotAccelerating(va.value)

an incomplete system implementation. All of the safety goals pertain to longitudinal acceleration and lateral steering. In this system, all features attempt to control acceleration, and LCA and PA attempt to control steering. CA, ACC, and LCA are only operational when the vehicle is in forward motion, RCA is only operational in reverse, and PA operates in both. Table 1 lists some of the safety goals identified.

An historic concern for automotive systems is unintended, or sudden acceleration [24]. Although most incidents of unintended acceleration have been attributed to driver error, as vehicles move closer to fully autonomous control, it becomes increasingly possible for such accelerations to be caused by system defects.

The first guard against unintended accelerations are safety goals S-01, which prohibits autonomous vehicle acceleration and autonomous vehicle jerk that exceed reasonable limits for comfort in vehicle motion [21]. The vehicle is, however, allowed to cause uncomfortable levels of negative acceleration (deceleration) when it performs sudden, uncomfortable stops to avoid collisions. Safety goal S-04 prohibits autonomous accelerations when the vehicle is stopped (e.g., if ACC stops vehicle behind a stopping lead vehicle or if CA stops the vehicle to avoid an object in the road). The driver is required to initiate acceleration by sending a "go" signal from the Human-Machine Interface (HMI), or by pressing the throttle pedal.

Another concern is feature interaction [6], which occurs when the combined behavior of two features is undefined in the specification of either. If one feature controls steering while another controls acceleration, a behavior may occur that is undefined by either, or by the system specification. Safety goal S-03 prohibits control of acceleration and steering by different subsystems if either is attempting to control both. It is possible, though, for the driver to override acceleration or steering. The system architecture includes an arbiter to select the control source of steering and acceleration. This use of an arbiter is a design choice of the developers and not necessarily required to meet the safety goals.

4.2. ICPA

Next, we applied ICPA to the system safety goals in Table 1 to identify subgoals for the subsystems. We limited the analysis to the five feature subsystems and the arbitration subsystem. Table 2 lists the safety subgoals for the arbiter (S-A-*X*) and features (S-F-*X*).

The goal coverage strategy for goals S-01, S-02, and S-04 is redundant responsibility. As the final source of system acceleration and steering commands, the arbiter becomes the primary source for meeting the system safety goal. The secondary safety goals of the feature subsystems provide protection against some single-point failures of the arbiter. If the arbiter fails by choosing an acceleration or steering command from the wrong feature subsystem source, then the system should still meet the safety goal. However, if the arbiter fails in a different way (e.g., summing requests from different features), then the features will not provide backup protection against the failure. The goal coverage strategy for goal S-03 is single responsibility, with the arbiter as the only source satisfying the goal. An alternative redundant responsibility strategy could require features to monitor other feature requests and cancel their own

Table 2. Safety subgoals for semi-autonomous automotive subsystems

Subsystem Safety Subgoals	
S-A-01	**Goal:** Achieve[AutoAccelCommandBelowThreshold] *(Arbiter)* **InformalDef:** *Acceleration commands caused by autonomous vehicle control shall not exceed* $2\ m/s^2$*.* **FormalDef:** $\forall$ ac: AccelerationCommand, ar: AccelerationRequest (ac.source = ●ar.source) $\Rightarrow$ (ac.value $\leq 2\ m/s^2$)
S-A-02	**Goal:** Achieve[AutoJerkCommandBelowThreshold] *(Arbiter)* **InformalDef:** *Acceleration commands caused by autonomous vehicle control shall not cause jerk to exceed* $2.5\ m/s^3$*.* **FormalDef:** $\forall$ ac: AccelerationCommand, ar: AccelerationRequest; (ac.source = ●ar.source) $\Rightarrow$ (VehicleJerkResponse(ac.value, ●ac.value) $\leq 2.5\ m/s^3$)
S-A-03	**Goal:** Achieve[SubsystemAccelSteeringCommandAgreement] *(Arbiter)* **InformalDef:** *If a subsystem requests control of the acceleration command and the steering command and is granted control of either the acceleration command or steering command, the subsystem shall control both acceleration and steering.* **FormalDef:** $\forall$ ac: AccelerationCommand, ar: AccelerationRequest, sc: SteeringCommand, sn: SubsystemName ●ar.active $\wedge$ ●sr.active $\wedge$ (●ar.source = ●sr.source) $\wedge$((ac.source = ●ar.source) $\vee$ (sc.source = ●sr.source)) $\Rightarrow$ (ac.source = sc.source = ●ar.source = ●sr.source) $\wedge$ (ac.value = ●ar.value) $\wedge$ (sc.value = ●sr.value)
S-A-04	**Goal:** Achieve[NoAutoAccelFromStop] *(Arbiter)* **InformalDef:** *If a) the vehicle is stopped for a duration of* StoppedTime *and b) the throttle pedal has not been applied within the preceding* AccelerationDelay *and c) a subsystem is controlling the acceleration command and d) the HMI has not sent a* go *signal to the controlling subsystem within the preceding* AccelerationDelay*, then the acceleration command shall be zero.* **FormalDef:** $\forall$ ac: AccelerationCommand, vsp: VehicleSpeed, sn:SubsystemName, hmi: HumanMachineInterface, tp: ThrottlePedal, st: StoppedTime, ad: AccelerationDelay ●■$_{<st}$IsStopped(vsp.value) $\wedge$ ¬●♦$_{<ad}$ Applied(tp) $\wedge$ (ac.source = sn) $\wedge$ ¬●♦$_{<ad}$ Applied(hmi.go, sn) $\Rightarrow$ NotAccelerating(ac.value)
S-F-01	**Goal:** Achieve[AutoAccelRequestBelowThreshold] *(CA, RCA, ACC, LCA, PA)* **InformalDef:** *Acceleration requests shall not exceed* $2\ m/s^2$*.* **FormalDef:** $\forall$ ar: AccelerationRequest; ❑(va.value $\leq 2\ m/s^2$)
S-F-02	**Goal:** Achieve[AutoJerkRequestBelowThreshold]*(CA, RCA, ACC, LCA, PA)* **InformalDef:** *Acceleration requests shall not cause jerk to exceed* $2.5\ m/s^3$*.* **FormalDef:** $\forall$ vj: VehicleJerk; ❑(VehicleJerkResponse(ar.value, ●ar.value) $\leq 2.5\ m/s^3$)
S-F-04	**Goal:** Achieve[NoAutoAccelRequestFromStop]*(CA, RCA, ACC, LCA, PA)* **InformalDef:** *If a) the vehicle is stopped for a duration of* StoppedTime *and b) the throttle pedal has not been applied within the preceding* AccelerationDelay *and c) the HMI has not sent a* go *signal to the controlling subsystem within the preceding* AccelerationDelay*, then acceleration requests shall be zero.* **FormalDef:** $\forall$ ar: AccelerationRequest, vsp: VehicleSpeed, sn:SubsystemName, hmi: HumanMachineInterface, tp: ThrottlePedal, st: StoppedTime, at: AccelerationTime ●■$_{<st}$IsStopped(vsp.value) $\wedge$ ¬●♦$_{<ad}$ Applied(tp) $\wedge$ ¬●♦$_{<ad}$ Applied(hmi.go, sn) $\Rightarrow$ NotAccelerating(ar.value)

when higher-priority features request control. However, this requires the arbitration logic to be built and maintained in multiple subsystems, which is impractical in this distributed development environment.

5. Evaluation and results

Although it is impossible to determine if the unknown/unrealizable part of the system safety goal has been minimized, it is possible to determine whether or not the subgoals produced by ICPA are useful. The purpose of goal elaboration is twofold. First, it is important to identify subsystem behaviors that contribute to hazards at design-time so that the subsystem development team can design and build the subsystem to help satisfy the system goal. Moreover, defining subsystem-level safety behaviors helps find potential problems early, before system integration. A second aim is to monitor the subsystem behaviors at run-time to determine whether a subsystem is violating its safety goals. Violations of subgoals may be predictors of system-level goal violations. In this work, the safety goals and subgoals were monitored in an implementation of the vehicle in a simulation environment.

5.1. Setup and evaluation scenarios

The vehicle system was partially implemented by a commercial automotive research lab as a Simulink® (version R2008a) model embedded in the CarSim® (version 7.01) simulation environment from Mechanical Simulation. CarSim® is a software system for simulating vehicle dynamic responses to different acceleration and steering inputs. For customized designs, models from other simulation environments such as Simulink® or Labview™ can be linked to the models.

Although the features were partially functional at the time of this evaluation, the implementation of the vehi-

Table 3. Safety goal violations for driving scenarios 1-3

		Simulation Time (s) / Duration (s)		
Goal	**Location**	**Scenario 1**	**Scenario 2**	**Scenario 3**
S-01	System	12.589/0.004	12.587/0.001	11 total first: 15.340/0.002, last: 15.471/0.010 longest: 0.062, shortest: 0.001
S-02	System	12.583/0.008, 12.598/0.002, 12.652/0.001, 12.661/0.004, 12.672/0.006, 12.680/0.001	12.568/0.020	79 total first: 12.492/0.004, last:15.492/0.004 longest: 0.036, shortest: 0.001
S-03	System		12.561/0.027	
S-04	System	*none*		
S-A-01	Arbiter	*none*		
S-A-02	Arbiter	12.601/0.001	12.561/0.001	22 total - first: 12.751/0.001, last: 15.401/0.001 longest: 0.001, shortest: 0.001
S-A-03	Arbiter		12.561/0.027	
S-A-04	Arbiter	*none*		
S-F-01		*none*		
S-F-02	CA	12.600/0.001		
	ACC			22 total - first: 12.750/0.001, last: 15.400/0.001 longest: 0.001, shortest: 0.001
	PA	9.623/0.001	9.623/0.001	9.623/0.001
S-F-04		*none*		

cle was incomplete. Safety goals were determined by examination of the functional requirements of the system, hazard analysis, and partial implementation.

In addition, the features were implemented independent of the process for defining and elaborating the system safety goals. That is, the formal specification of the safety goals and subgoals was not available to the teams designing and implementing the system. As such, the purpose of this evaluation was not to demonstrate that ICPA helped in system design and implementation, but rather to determine whether the goals were partially, if not fully, composed by the set of subgoals obtained from ICPA. To do this, we monitored the system safety goals and subgoals at run-time in three different driving scenarios, configured in CarSim®.

In the first scenario, the host vehicle is accelerating from zero velocity toward another vehicle that is stopped in its path. As the host vehicle nears the stopped vehicle, CA should initiate a hard braking action to stop the host vehicle before a collision occurs.

The second scenario proceeds as the first, except just as CA is stopping the host vehicle, PA is engaged by the driver. This scenario was chosen to verify a flaw in the arbitration logic found during ICPA. In the design and implementation, acceleration and steering were arbitrated separately. During inspection for ICPA we discovered that prioritization of feature requests in steering arbitration was reversed.

In the third scenario, ACC is engaged and controlling the host vehicle at a set speed, with a slower vehicle in its path. ACC should slow the host vehicle to follow a set distance behind the slower lead vehicle.

5.2. Results

Table 3 lists the results of the three evaluation scenarios. The first evaluation scenario, intervention by CA to stop the vehicle, resulted in violation of vehicle safety goals S-01 and S-02 shortly before early termination of the simulation. The longest violation was 8 ms, and the shortest was 1 ms (the time interval of one state). Vehicle acceleration was exceeded for 4 ms, at 92 ms before early termination of the system. Vehicle jerk was exceeded for 8, 2, 1, 4, 6, and 1 ms, at 98, 83, 29, 20, 9, and 1 ms before early termination of the system. Although the vehicle acceleration threshold was exceeded once, no violations of S-A-01 and S-F-01 occurred. Similarly, the vehicle jerk threshold was exceeded six times, yet the corresponding subgoals for the arbiter and CA were violated only once for 1 ms, starting 80 ms before early termination of the system. In addition, the jerk threshold for PA was violated for 1ms at 3.058 seconds before early termination of the system.

The second evaluation scenario resulted in violation of vehicle safety goals S-01, S-02 and S-03, also before early termination of the simulation. In this situation, vehicle acceleration was exceeded for 1 ms, vehicle jerk was exceeded for 20ms, and the acceleration-steering agreement was violated for 27 ms. Each vehicle safety goal was still in violation at the time of early termination, which occurred 93 ms earlier than early termination in the first scenario. Similar to the first scenario, no subgoals for S-01 were violated, and S-A-02 for the arbiter was violated only once for 1 ms, starting 27 ms before early termination of the system and 7 ms before vehicle jerk was exceeded. As in the first scenario, the corresponding subgoal S-F-02 for PA was violated for 1ms, starting 80ms before early termination of the system. The S-A-03 subgoal for the arbiter was the same as the system-level goal, thus violations of the arbiter's goal corresponded to those of the system.

In the third scenario, the vehicle exceeded the acceleration threshold 11 times, and the jerk threshold 79 times, before early termination of the simulation. As expected, ACC attempted to slow the host vehicle to follow the slower lead vehicle. However, the host vehicle was not slowed enough, and eventually passed through the lead vehicle. (In CarSim, crashes with other vehicles are not simulated. However, crashes due to host vehicle instability are.) As with the first two scenarios, the violations of S-01 did not correspond to any violations of its corresponding S-A-01 and S-F-01. In addition, the arbiter and ACC experienced 29 violations of S-A-01 and S-F-01, respectively, which is much fewer than the number experienced by the system.

Although the subsystem monitors did not identify all events in which the system safety goals were violated, the results of simulation do provide some useful information. Monitoring at the system and subsystem level identified violations of the safety goals and subgoals that may be imperceptible to system testers, who may not notice if the jerk and acceleration thresholds are violated for just a few short milliseconds. Although a very short interval of uncomfortable jerk or acceleration may not startle the driver or lead to an accident, it may portend future trouble. All three scenarios ended in early termination of the simulation due to a simulation runtime error. In the first two scenarios, this early termination was preceded by very short intervals of violations of safety goals S-01 and S-02, and subgoals S-A-02 and S-F-02. In addition, we were able to verify the reverse arbitration logic found during ICPA (which resulted in violation of safety goal S-03) as being incorrect.

The violations of vehicle jerk by CA and the Arbiter subgoals occurred for 1ms, which is just one state in the transition system. In some situations, a violation of this type is a false positive because it occurs whenever a step increase in acceleration occurs. In these scenarios, however, violation of jerk subgoals indicated unsafe vehicle conditions because CA suddenly released and reapplied the hard brake prematurely before the system had finished braking to avoid colliding with the parked vehicle. Because the system implementation was incomplete, it is unknown whether or not this is the only cause of early termination of the simulation. However, it is most likely a contributing, if not primary, factor.

The violation of the jerk subgoal for PA in all three scenarios technically represents a false positive detection in the subgoal monitors. The sudden jerk in requested acceleration did not result in a violation of the vehicle jerk threshold. However, this detection does expose a potential problem in the PA subsystem. At simulation time 9.330 PA requests an acceleration of -2 m/s^2 without indicating that its acceleration request is active. At simulation time 9.622 its acceleration request returns to 0 m/s^2, the default value for all subsystems when no request is active. In other words, it changes the value of its acceleration request without signaling that the request is active. It is unclear whether PA intended to make a parking maneuver and failed, or whether it is sending random acceleration requests. In either situation, the subsystem must be examined to determine the cause of the unsafe behavior.

The subgoal monitors did produce false negatives when the vehicle acceleration and vehicle jerk thresholds were exceeded without a violation in the corresponding subgoal, such as what occurs in the first scenario. In these instances, vehicle acceleration and jerk were under control of one of the subsystems, but no subsystem monitor indicated the subsystem was violating the subgoal. This indicates the system-level goal has not been fully decomposed. In this situation, the remaining emergence most likely represents missing subgoals for other system components, such as the powertrain and braking systems that were excluded from the analysis. This may also indicate that some relationships between indirect control sources are missing from the ICPA. These safety problems can still be handled by system-level safety goals. Our finding is simply that they were not completely decomposed.

6. Conclusion and future work

When dealing with emergent properties, it may not be possible to fully elaborate system-level goals throughout the composite system. However, it is important to identify subgoals for the known sources of the emergent behavior, even if other sources are unknown or impractical to address. Identifying subsystem-level safety issues can reduce the number and severity of problems found during system integration, and perhaps give precursor warnings at run-time.

In this work, we define emergent and composable behaviors mathematically, in the context of system safety goals written as expressions of temporal logic and the ICPA technique. ICPA was applied to a safety-critical vehicle system designed and built in a commercial automotive research lab, and the resulting safety goals and subgoals were monitored in different run-time operational scenarios. Although this system was intended only for research, its design and implementation are as complex as production vehicles. Results of monitoring of the goals and subgoals in run-time simulations show that although the subgoals do not fully compose the parent goals, they are useful for identifying some violations of the parent goal. In addition, monitoring safety goals in general, at both the subsystem and system level, identified potentially hazardous behaviors that may be imperceptible to system testers.

This work demonstrates the feasibility of using ICPA to identify subgoals that partially compose an emergent safety goal. Future work will focus on determining how much composability can be achieved with ICPA. The run-time monitoring scenarios all experienced system failures from existing system flaws, due to the system being incomplete at the time of analysis. Fault-injection campaigns on a more complete implementation might provide a statistical measure of composability extracted by ICPA from an emergent property.

References

[1] Aristotle. *Metaphysics: Book VIII*. Trans. by W.D. Ross, http://classics.mit.edu/Aristotle/metaphysics.8.viii.html.

[2] L. Bass, P. Clements, and R. Kazman. *Software Architecture in Practice*. Addison-Wesley, Reading, MA, 2nd edition, 1972.

[3] F. Bitsch. Classification of safety requirements for formal verification of software models for industrial automation systems. In *Proc. Intl. Conf. on SW and Sys. Engineering and their Applications (ICSSEA 2000)*, Paris, France, Dec. 2000.

[4] F. Bitsch. Safety patterns - the key to formal specification of safety requirements. In U. Voges, editor, *Proc. 20th Intl. Conf. on Comp. Safety, Reliability and Security (SAFECOMP '01)*, volume 2187 of *Lecture Notes in Computer Science*, pages 176–189, Budapest, Hungary, Sept. 2001. Springer-Verlag.

[5] J. Black and P. Koopman. Indirect control path analysis and goal coverage strategies for elaborating system safety goals in composite systems. In *Proc. 14th Pacific Rim Intl. Symp. on Dependable Comp. (PRDC'08)*, Taipei, Taiwan, Dec. 2008.

[6] T. F. Bowen, F. S. Dworack, C. H. Chow, N. Griffeth, G. E. Herman, and Y.-J. Lin. The feature interaction problem in telecommunications systems. In *Proc. 7th Intl. Conf. on SW Eng. for Telecom. Switching Sys.*, pages 59–62, Bournemouth, UK, July 1989.

[7] P. Cariani. Emergence and artificial life. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, editors, *Artificial Life II: Proc. of the 2nd Workshop on Artificial Life*, pages 775–797. Addison-Wesley, 1991.

[8] A. Dardenne, A. v. Lamsweerde, and S. Fickas. Goal-directed requirements acquisition. In M. Sintzoff, C. Ghezzi, and G. Roman, editors, *Science of Computer Programming*, number 1-2, pages 3–50. Elsevier Science, Amsterdam, The Netherlands, Apr. 1993.

[9] R. Darimont and A. v. Lamsweerde. Formal refinement patterns for goal-driven requirements elaboration. In *Proc. 4th ACM SIGSOFT Symp. on Found. of SW Eng.*, volume 21 of *Software Engineering Notes*, pages 179–190. ACM SIGSOFT, Nov. 1996.

[10] V. Darley. Emergent phenomena and complexity. In R. A. Brooks and P. Maes, editors, *Artificial Life IV: Proc. of the 4th Workshop on Synthesis and Simulation of Living Systems*, pages 411–416. MIT Press, 1994.

[11] M. S. Feather. Language support for the specification and development of composite systems. *ACM Trans. on Prog. Lang., Syst.*, 9(2):198–234, Apr. 1987.

[12] S. E. Keller, L. G. Kahn, and R. B. Panara. Specifying software quality requirements with metrics. In R. H. Thayer and M. Dorfman, editors, *System and Software Requirements Engineering*, chapter 3, pages 145–163. IEEE Comp. Soc. Press, Los Alamitos, CA, 1990.

[13] R. Koymans, editor. *Specifying Message Passing and Time-Critical Systems With Temporal Logic*, volume 651 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, Germany, 1992.

[14] A. v. Lamsweerde, R. Darimont, and E. Letier. Managing conflicts in goal-driven requirements engineering. *IEEE Trans. SW Eng.*, 24(11):908–926, Nov. 1998.

[15] E. Letier and A. v. Lamsweerde. Agent-based tactics for goal-oriented requirements elaboration. In *Proc. 24th Intl. Conf. on SW Eng. (ICSE'02)*, pages 83–93, Orlando, FL, May 2002.

[16] E. Letier and A. v. Lamsweerde. Deriving operational software specifications from system goals. In *Proc. 10th ACM SIGSOFT Symp. on Foundations of SW Eng. (FSE-10)*, pages 119–128, Charleston, SC, Nov. 2002.

[17] N. G. Leveson. *Safeware - System Safety and Computers*. Addison-Wesley, Reading, MA, 1995.

[18] N. G. Leveson. Intent specifications: An approach to building human-centered specifications. *IEEE Trans. SW Eng.*, 26(1):15–35, Jan. 2000.

[19] N. G. Leveson. Model-based analysis of socio-technical risk. Technical Report ESD-WP-2004-08, MIT, Cambridge, MA, Dec. 2004.

[20] G. H. Lewes. *Problems of Life and Mind*, volume 2 of *1st Ser.* James R. Osgood and Co., Boston, MA, 1875.

[21] P. Li, L. Alvarez, and R. Horowitz. Ahs safe control laws for platoon leaders. *IEEE Trans. Contr. Syst. Technol.*, 5(6):614–628, Nov. 1997.

[22] J. Mylopoulos, L. K. Chung, and B. A. Nixon. Representing and using nonfunctional requirements: A process-oriented approach. *IEEE Trans. SW Eng.*, 18(6):483–497, June 1992.

[23] N. J. Nilsson. *Problem-Solving in Artificial Intelligence*. McGraw-Hill Computer Science Series. McGraw-Hill, New York, 1971.

[24] J. Pollard and E. D. Sussman. An examination of sudden acceleration. Technical Report DOT-HS-807-367, NHTSA, Washington, DC, 1989.

[25] M. Privosnik, M. Marolt, A. Kavcic, and S. Divjak. Evolutionary construction of emergent properties in multi-agent systems. In *Proc. 11th Mediterranean Electrotechnical Conference (MELECON'02)*.

[26] A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proc. of the London Mathematics Society*, 42:230–265, 1936.