# S-PDH: A CPS Service Contract Framework for Composition

Lin Ye, Kaiyu Qian, and Liang Zhang(✉)

Shanghai Key Laboratory of Data Science, School of Computer Science,
Fudan University, Shanghai, China
{11110240021,14210240018,lzhang}@fudan.edu.cn

**Abstract.** Cyber-Physical Systems (CPSs) are complex systems in which physical processes are tightly interacting with networked computing components. Some pilot researches suggest that Service-Oriented Architecture (SOA) by promising to guide the development of CPSs, but most of them neglect continuous physical behaviors for coinciding with traditional SOA. Consequently, developed CPS services cannot properly support the tight interaction between physical processes and computing. In this paper, we propose a novel framework S-PDH, in which a CPS service is characterized by a three-level of service contracts, namely, the *physical property contract*, the *dynamic physical behavior contract*, and the *hybrid system behavior contract*. Based on the framework, we study CPS service composition. This study introduces a novel SOA solution to CPS development, and promotes service computing to a new frontier.

**Keywords:** Service-Oriented Architecture (SOA) · Cyber-Physical System (CPS) · Modeling · Service contract · Service composition

## 1 Introduction

Cyber-Physical Systems (CPSs) are emerging systems that will reshape the way our modern society perceives the physical world, lives, moves, and interaction in it [1]. In CPSs, computers monitor and control the physical processes via networks in the manner of feedback loops where physical processes affect computations as well [2]. Due to the historical ties to embedded systems, most CPSs so far are designed as *monolithic* ones. The situation gravely impedes CPSs reaching more broad and complex territory. CPSs are seeking more effective paradigms to leverage their development [1].

Meanwhile, Service-Oriented Computing (SOC) has been proved to be an elegant paradigm to deal with complex distributed systems [3]. Corresponding Service-Oriented Architecture (SOA) can guide us to integrate heterogeneous components and make the system scale upwards easily. Therefore, SOC is a promising paradigm to bail the CPS out.

In fact, there have been many studies about CPS services. Researchers have analyzed physical properties [4–6], studied device behaviors [7,8], or probed time-spatial features [9,10]. However, most of them neglect continuous physical behaviors for coinciding with traditional SOA. This compromise expressiveness or verification capability. In CPS, as we know, improper treatment of physical behaviors might lead to low efficiency, serious functional errors or even disasters.

To fill the gap, we propose a method to characterize CPS services, and leverage their composition, in accordance with the roadmap of SOC research [3]. Physical behaviors in CPS are described by three successive levels, from bottom to top, physical properties, dynamic physical behaviors, to hybrid system behaviors.

Our technical contributions include:

- a framework S-PDH which provides a comprehensive model to characterize CPS intrinsic features, and supports CPS services composition;
- the capability of precise control of efficiency and safety in CPSs;
- leveraging the SOC to deal with interacting dynamic systems with both discrete and continuous behaviors.

The paper is organized as follows. Section 2 identifies the necessity of physical behavior characterization and the compositional construction of CPSs via a running example. Section 3 sketches the S-PDH framework, and highlights the rationale underpinning our idea. Section 4 discusses composition facilities in S-PDH. Section 5 outlines the prototype implementation and a case study. In Sect. 6 we review related work. Finally, we conclude the study in Sect. 7.

## 2 Motivating Example

Let us consider a smart crossing scenario. In its simplest setting, there is a one-way street with a crosswalk (Fig. 1(c)). There are three kinds of participants: cars, traffic lights, and pedestrians. An intent CPS service, named SmartCrosswalk, tries to meet two goals: efficiency and safety. By efficiency, we mean it should let cars and pedestrians pass the crossing as many as possible. By safety, we must be sure that there is no any collision of cars or pedestrians.

In this context, some design issues are:

- How to model participants and their interactions effectively? Can we improve the system performance with the help of the dynamic physical behavior such as cars acceleration?
- Can we develop scalable facilities to support large-scale CPS services, e.g. constructing complex CPS services like Fig. 1(d), or (e) effectively and efficiently by composition?

CPS service approaches so far solve the problem 2 partially, but fail to handle the problem 1. In a word, the CPS services cry for innovations.

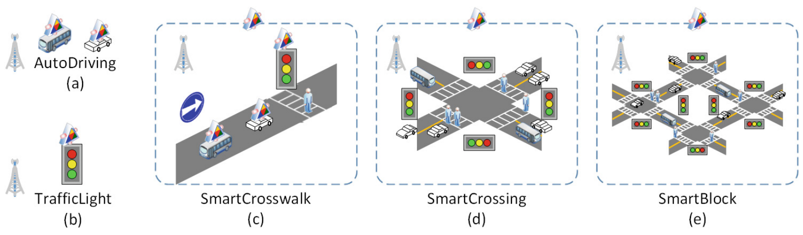


**Fig. 1.** Constructing complex CPSs progressively. (a) The AutoDriving CPS, (b) The TrafficLight CPS, (c) Constructing SmartCrosswalk from the AutoDriving and the TrafficLight, (d) Constructing SmartCrossing from the SmartCrosswalk, and (e) Constructing SmartBlock from the SmartCrossing or the SmartCrosswalk

## 3 The S-PDH Framework

In order to capture rich features in CPSs, we propose the S-PDH framework as Fig. 2, which characterizes CPS services progressively through physical properties, to dynamic physical behaviors, until hybrid system behaviors, in accordance with the service foundation dimension [3].

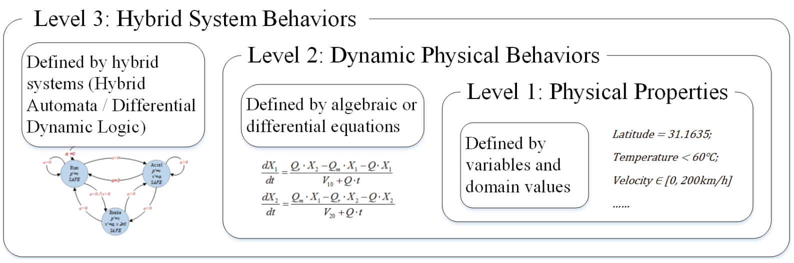

Level 3: Hybrid System Behaviors
Defined by hybrid systems (Hybrid Automata / Differential Dynamic Logic)
Level 2: Dynamic Physical Behaviors
Defined by algebraic or differential equations
$\frac{dX_1}{dt} = \frac{Q \cdot X_2 - Q_m \cdot X_1 - Q \cdot X_1}{V_{10} + Q \cdot t}$
$\frac{dX_2}{dt} = \frac{Q_m \cdot X_1 - Q_r \cdot X_2 - Q \cdot X_2}{V_{20} + Q \cdot t}$
Level 1: Physical Properties
Defined by variables and domain values
Latitude = 31.1635;
Temperature < 60°C;
Velocity ∈ [0, 200km/h]
......

**Fig. 2.** S-PDH, a three level framework to characterizing CPS services

### 3.1 Modeling Physical Properties

Physical properties are captured by property variables that reflect the snapshots of physical behaviors of a CPS. The variable range, normally in $\mathbb{R}$, represents physical status, performance, etc. Basically, a physical property is a tuple

$$\alpha_{l1} ::= \langle pname, ptype, pvtype, pvcons, pvunit \rangle,$$

where:

- *pname*: string, the physical property name;
- *ptype*: string, taking from a set of predefined types, e.g. time, dimension, position, energy, temperature, etc.;
- *pvtype*: string, the physical property value type, it can be a simple type such as integer, float, double, boolean, string, etc. or complex type defined by schema such as ranging in $\mathbb{R} \times \mathbb{R}$;

- *pvcons*: the physical property constants or constraint values corresponding to pvtype;
- *pvunit*: the unit of physical property measurement, which is important for unit matching.

Physical properties are supported by traditional CPS service approaches [4–10]. But this facility alone cannot support CPS effectively. Taking the CPS AutoDriving (Fig. 1(a)) as an example, the velocity of a car could be as high as 150 km/h, but if there is another car ahead, the braking distance will impose restriction on the real velocity, e.g. at most 70 km/h at certain moment. Hence, in order to describe physical behaviors precisely, it is required to characterize richer features in CPS service contracts, following SOA priniples.

## 3.2 Modeling Dynamic Physical Behaviors

Most physical behaviors can be represented formally by the combination of differential and algebraic equations [11]. In our running example, cars running and acceleration behaviors can be expressed by

$$\dot{s} = v, \dot{v} = a. \tag{1}$$

where $s$ represents the passage within time $t$ with the velocity $v$, and $a$ is the acceleration.

The dynamics of the physical components is an equation of time $t$. We use linear differential equations and linear algebraic equations, i.e. ako linear time invariant (LTI) [11] to describe all dynamic physical behaviors based on modern control theory. To do so, we have

- **Input Variables**: the set of variables that describe the inputs of the dynamics, e.g. $a$ in the Eq. (1).
- **Controlled Variables**: the set of variables that describe the result of the dynamics, e.g. $s, v$ in the Eq. (1).
- **Output Variables**: the set of controlled variables that we are pursuing, e.g. $s$ in the Eq. (1). We may call the rest of controlled variables **Intermediate Variables**, e.g. $v$ in the Eq. (1).

Now a physical dynamic system can be modeled as Fig. 3.

Generally, the model of a system is a function in the form [11]

$$F : X \to Y, X = Y = \mathbb{R}^{\mathbb{R}} \tag{2}$$

To conclude, the dynamic physical behavior description is a tuple

$$\alpha_{l2} ::= \langle inpvar, intvar, outvar, func \rangle,$$

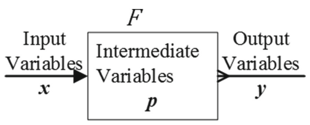


**Fig. 3.** A physical dynamic system

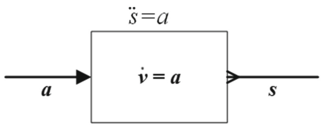


**Fig. 4.** The car's acceleration behavior

where:

- the first three elements represent input variables, internal variables and output variables, respectively. Note that each element in the tuple is still a tuple in the form $\langle pname, ptype, pvtype, pvcons, pvunit\rangle$, and each variable could be a function of time $t$;
- $func$: the algebraic or differential equations of the variables.

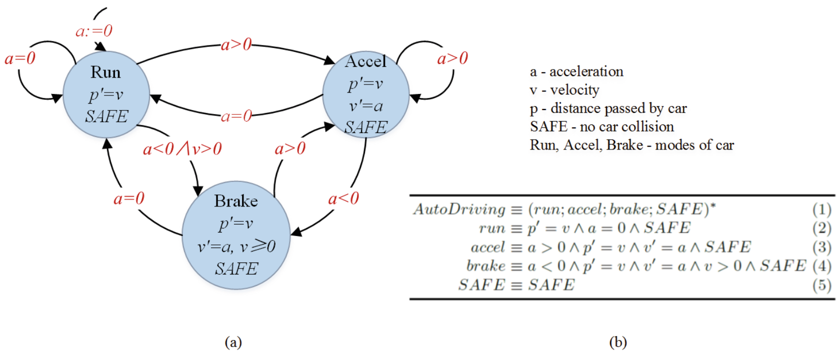
As an example, for the Eq. (1), its model is depicted as Fig. 4.

## 3.3 Modeling Hybrid System Behaviors

The intertwining of discrete and continuous dynamics is the intrinsic feature of a CPS [12]. An excellent formal model to support the intertwining is the so-called *hybrid system*, in which there are system flows (by differential equations) and jumps (by difference equations) [13]. We will use hybrid automaton or hybrid program [14] interchangeably for intuitive or expressive power purposes.

In this way, the hybrid system can be modelled as a hybrid automaton or a hybrid program. Furthermore, we choose Differential Dynamic Logic (d$\mathcal{L}$) [14] to characterize the hybrid system behavior.

Example 1. The hybrid behavior of an AutoDriving CPS service for the auto-driving of multiple cars is depicted by Fig. 5.



$$AutoDriving \equiv (run; accel; brake; SAFE)^* \quad (1)$$
$$run \equiv p' = v \wedge a = 0 \wedge SAFE \quad (2)$$
$$accel \equiv a > 0 \wedge p' = v \wedge v' = a \wedge SAFE \quad (3)$$
$$brake \equiv a < 0 \wedge p' = v \wedge v' = a \wedge v > 0 \wedge SAFE \quad (4)$$
$$SAFE \equiv SAFE \quad (5)$$

**Fig. 5.** Hybrid automaton (a) and hybrid program (b) for AutoDriving hybrid system

Accordingly, we have

**Hybrid Programs** [14]:

$$\alpha, \beta ::= x_1 := \theta_1, \cdots, x_n := \theta_n | x_1' := \theta_1, \cdots, x_n' := \theta_n \& \chi | ?\chi | \alpha \cup \beta | \alpha ; \beta | \alpha^*$$

where $\alpha, \beta$ are hybrid programs, $\theta_i$ are $\mathsf{d}\mathcal{L}$ terms, $x_i \in \Sigma$ are state variables, and $\chi$ is a hybrid formula of first-order logic over reals.

Here, the $\mathsf{d}\mathcal{L}$ terms and hybrid formula are defined as:

$\mathsf{d}\mathcal{L}$ **Terms** [14]:

$$\theta ::= x | f(\theta_1, \cdots, \theta_n).$$

where $\theta_1, \cdots, \theta_n$ are terms, $f$ is a function symbol of arity $n$, and $x \in \Sigma$ is a logical variable.

**Hybrid Formulas** [14]:

$$\phi, \psi ::= p(\theta_1, \cdots, \theta_n) | \neg\phi | (\phi \wedge \psi) | (\phi \vee \psi) | (\phi \rightarrow \psi) | \forall x \phi | \exists x \phi | [\alpha]\phi | \langle\alpha\rangle\phi$$

where $\phi, \psi$ are $\mathsf{d}\mathcal{L}$ formulas, $\theta_i$ are terms, $p$ is a predicate symbol of arity $n$, $x \in V$ is a logical variable, and $\alpha$ is a hybrid program.

Note that $\alpha \cup \beta | \alpha ; \beta | \alpha^*$ in the hybrid programs definition are composition of hybrid programs, which will be used in the next section.

Then, the hybrid system behaviors description in S-PDH is the tuple

$$\alpha_{l3} ::= \langle HybridPrograms \rangle.$$

It is not obligated that every CPS service populates all its three levels of contracts. We could choose suitable S-PDH levels in application to fit specific contexts so long as it meets the requirement that the higher level depends on the lower levels.

## 4 CPS Service Composition in S-PDH

Along another dimension of SOC research roadmap in [3], we use composition as the means to develop new coarser grained services for value-added purpose.

**CPS Service Composition:** A composition of CPS services is a pair $\langle \mathcal{C}, \mathcal{U} \rangle$, standing for the constitution part and the utility part, respectively, where,

$$\mathcal{C} :== \alpha | \mathcal{C} || \mathcal{C}, \qquad \mathcal{U} :== p(\mathcal{C}) | v(\mathcal{C}) | d(\mathcal{C}) | h(\mathcal{C}) | s(\mathcal{C}).$$

The constitution part reflects the fact that the composition is closed, i.e. the composite service is still a CPS service. More specifically,

- $\alpha$ is a triple $\langle \alpha_{l1}, \alpha_{l2}, \alpha_{l3} \rangle$, where elements $\alpha_{l1}, \alpha_{l2}$, and $\alpha_{l3}$ are defined in the last section, along the S-PDH framework;
- $\mathcal{C} || \mathcal{C}$ represents the parallel composition of two CPS services who are mutually compatible at the same level in S-PDH.

On the other hand, the utilities about $\mathcal{C}$ have following meaning:

- $p(\mathcal{C})$ checks compatibility on physical properties;
- $v(\mathcal{C})$ takes actions on variables. Three operations are: recognizing shared variables, initializing variables function, and evaluating variables function;
- $d(\mathcal{C})$ takes actions on dynamic physical behaviors. There are two important operations: initializing dynamic physical behavior function, and evaluating dynamic physical behavior function.
- $h(\mathcal{C})$ takes actions on hybrid system behaviors. There are two important operations: recognizing hybrid system behavior states, and evaluating hybrid system behavior states.
- $s(\mathcal{C})$ takes the synchronization actions as needed. There are four operations: synchronizing physical properties, synchronizing variables, synchronizing dynamic physical behaviors, and synchronizing hybrid system behavior states.

In composing CPS services, we need to guarantee the compatibility of contracts at all the three levels.

## 4.1 Compatibility Checking for Physical Properties

At the property level, i.e. $\alpha_{l1}$,

- for $\mathcal{C}||\mathcal{C}$, we analyze the intersection of variable domains to compose two physical properties without resource contention. We could use $p(\mathcal{C})$ and $s(\mathcal{C})$ to conquer the nondeterministic;
- $p(\mathcal{C})$ evaluates physical properties, which can be $pvtype, pvcons$, and $pvunit$.
- $v(\mathcal{C}), d(\mathcal{C})$, and $h(\mathcal{C})$ are not available at this level;
- $s(\mathcal{C})$ takes the synchronization operations on physical properties at this level.

**Example 2.** For the AutoDriving case in Example 1, the car's location is at *latitude = 31.1635144, longitude = 121.579742*. So the following car can check whether the location conflicts if it is going to get that place.

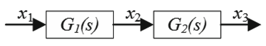
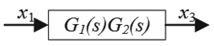
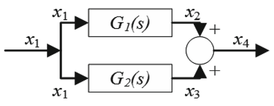
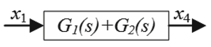
## 4.2 Compatibility Checking for Dynamic Physical Behaviors

We use the control theory [11], i.e. transfer functions and block diagram models, to develop composition analysis rules as Table 1.

Based on these rules, we propose an algorithm (Algorithm 1 below) to check the compatibility at the dynamic physical behavior level without any information lost. More specifically, the $\mathcal{C}||\mathcal{C}$ represents the parallel composition of two $\alpha_{l2}$.

- $v(\mathcal{C})$ takes the steps 1–8 of Algorithm 1.
- $d(\mathcal{C})$ takes the steps 9–17 of Algorithm 1.
- $h(\mathcal{C})$ is not available at this level.
- $s(\mathcal{C})$ takes actions on demand.

**Table 1.** Transformation with equivalent diagram

| Transformation | Original Diagram | Equivalent Diagram |
|---|---|---|
| 1. Cascade Combining |  |  |
| 2. Parallel Combining |  $x_1$<br>$x_1$<br>$G_1(s)$<br>$x_2$<br>+<br>$x_4$<br>$x_1$<br>$G_2(s)$<br>$x_3$<br>+ |  |

**Algorithm 1.** Composition analysis for dynamic physical behaviors analysis

```
Input: $\langle inpvar1, intvar1, outvar1, func1\rangle, \langle inpvar2, intvar2, outvar2, func2\rangle$
Output: $ComposedFunc$
 1: if $(inpvar1 \cup intvar1 \cup outvar1) \bigcap (inpvar2 \cup intvar2 \cup outvar2) == \emptyset$ then
 2:     return false
 3: else
 4:     $SharedVar = (inpvar1 \cup intvar1 \cup outvar1) \bigcap (inpvar2 \cup intvar2 \cup outvar2)$ {Check for
        common variable domain area}
 5:     if $Evaluate(SharedVar) == \varnothing$ then
 6:         return false
 7:     end if
 8: end if
 9: $TranFunc1 = Laplace(func1), TranFunc2 = Laplace(func2)$
10: if $Typeof(SharedVar) == [inpvar1, inpvar2]||[intvar1, intvar2]||[outvar1, outvar2]$ then
11:     $ComposedTransFunc = TranFunc1 + TranFunc2$
12: else if $Typeof(SharedVar) == [inpvar1, intvar2]||[inpvar1, outvar2]||[inpvar1, outvar2]$
    then
13:     $ComposedTransFunc = TranFunc1 \cdot TranFunc2$
14: end if
15: $ComposedFunc = UnLaplace(ComposedTransFunc)$
16:
17: return $ComposedFunc$
```

**Example 3.** In Example 1, the AutoDriving car has a brake-distance function $brakedistance = \frac{v^2}{2g\mu}$. However, it is affected by current velocity and road condition. We now can get a more compact safe distance between two cars by evaluating their brake-distance, much better than other approaches that merely use the physical properties, e.g. *max brake-distance*.

### 4.3 Compatibility Checking for Hybrid System Behaviors

With the help of $\mathsf{d}\mathcal{L}$ [14], we can check the compatibility of hybrid system behaviors systematically. More specifically, hybrid programs form a regular-expression-style Kleene algebra with tests. Along this line, $\mathcal{C}||\mathcal{C}$ represents the parallel composition of two $\alpha_{l3}$, which is supported by $\alpha \cup \beta, \alpha; \beta, \alpha^*$.

- $p(\mathcal{C}), v(\mathcal{C}), d(\mathcal{C})$ are checked as above;
- $h(\mathcal{C})$ will conduct the state reachable analysis for hybrid systems, and system condition verification, etc.
- $s(\mathcal{C})$ is in charge of synchronizing the variables, states in certain conditions.

In our study, we use KeYmaera [15] to verify the hybrid system behaviors.

## 5 Prototype Implementation and Evaluation

To validate the feasibility and the effectiveness of the S-PDH framework, we implement a prototype around the motivating example in Sect. 2.

We use JAX-WS in SOAP style, and extend ⟨ComplexType⟩ in WSDL 2.0 to support the three level contracts in S-PDH. We also use Mathematica as a differential equation solver to meet the requirements of the $\mathsf{d}\mathcal{L}$ with the KeYmaera [15].

We choose JAX-WS instead of more sophisticate WS-* standards because it is a lightweight solution to Web Services and is bundled within JDK.

To validate the feasibility of the S-PDH, we conduct a case study against the motivating scenarios. The atomic CPS services TrafficLight and AutoDriving are straightforward as depicted in above examples.

What's really interesting is that we can compose CPS services to a new CPS service, and let the process progress iteratively. Note the corresponding in S-PDH between a CPS service and a CPS, this capability implies a new approach for CPS development. For example, the CPS service SmartCrosswork (Fig. 1(c)) can be constructed by two autonomous CPS services, say AutoDriving and TrafficLight, by integrating all the three levels of contracts as Fig. 6.

Based on the SmartCrosswalk service, we can further get SmartCrossing (Fig. 1(d)) and SmartBlock Service (Fig. 1(e)) by composition such as Fig. 7.

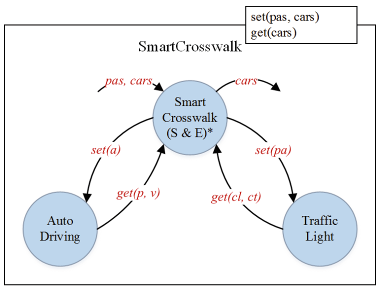


**Fig. 6.** SmartCrosswalk service

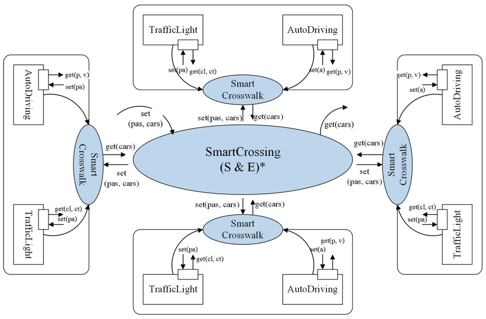


**Fig. 7.** SmartCrossing service

To make our approach more concrete, we apply it on digilent's ZRobots. By embedding our prototype into the ZedBoard (running ARM-based Linux), we get AutoDriving Service. Please refer to [16] for implementation details. The test shows that by acquiring the motion equations of front car provided by its AutoDriving service, the following car can evaluate the distance change curve with fully braking by Mathematica, which demonstrates the precise control of safety and efficiency with S-PDH by introducing physical behaviors in service contracts.

## 6 Related Work

Over last five years, more and more researchers have pursued SOA-based CPSs. We review them by categories.

***Contract-Based Services Modeling.*** Contract-based services modeling was introduced in [17], and developed by [18]. In [19], the author claimed that it will bring huge advantages of exploiting behavioural information for service discovery and composition, which inspires us to utilize the contract-based services modeling approach in constructing CPS services.

***Physical Resource / Physical Property Focused Model.*** Typical SOA-based extensions for CPS take into consideration of the physical resource constrains, which differs from traditional software services [20–23]. Physical property model utilizes the semantic methods [4,5] or context model [6] to examine the compatibility of different physical properties. They evolve the service constrains with physical resources and some related physical properties, and utilize QoS to fulfill the requirement of CPS service.

Our work encloses the physical resources and physical properties into S-PDH Level 1, and focuses on more comprehensive physical behaviors at higher abstraction levels, i.e. the Level 2, and the Level 3.

***Virtual Device Operating Methods.*** Some researchers study how to capture device operating and results [7,8]. They wrap the physical part as virtual devices and transform the device operating as service event/control process.

Our work embeds the event/process control into the dynamic physical behaviors (Level 2 in S-PDH). We go further by introducing hybrid system behaviors, which leverages CPS services to more complex environments.

***Time-Spatial Extension Methods.*** Time-spatial constraints are vital ingredients in CPSs. So some researchers focus on how to extend the capacity of time-spatial handling in SOA. For example, [9,10] utilize time-space $\pi$-calculus or real-time control middleware to operate the resource, time and space constraints of CPSs.

As we discussed in the motivating example, only time-spatial constraint is not enough to keep the system's efficiency and safety. Our work considers the physical properties as well as behaviors at various levels. So we get a more powerful methods to handle the efficiency and safety requirements.

***Hybrid System Extension Methods.*** Our former work [24,25] proposed a CPS service extension method based on hybrid system, which can model the system physical behavior well. They merely focused on compatibility verification instead of the comprehensive framework here. Our recent work [16] focuses the implementation rather than the framework. They can be considered as the supplement of this article.

## 7 Conclusion and Future Work

We develop a framework S-PDH that provides a comprehensive model to develop CPS services. First, by casting physical behaviors into service contracts, a service consumer is able to anticipate the physical state of the service provider dynamically based on the physical process pattern, which will contribute a lot to precise control of conflict goals in CPS. Then, we leverage the contract notion to implement CPS service composition, which results in a scalable facility to support large-scale CPS construction.

On the other hand, with the physical behaviors contract extension, SOC is capable of dealing with the interacting dynamic systems with both discrete and continuous behaviors. This might leverage SOA technology to a new frontier.

There are still many research issues around the proposed framework, e.g. service compatibility analysis, model checking, and automatic compositing about CPS services. We will work on them in the future work.

**Acknowledgments.** The work is partially supported by NSFC (No.60873115), MoE & China Mobile Joint Foundation (No.MCM20123011), Shanghai S & T Funds (No.13dz2260200 & No.13511504300), National Hi-tech Project ( No. 2012AA02A602).

## References

1. Poovendran, R., et al.: Special issue on cyber - physical systems. Proc. IEEE **100**, 6–12 (2012)
2. Lee, E.A., Seshia, S.A.: Introduction to Embedded Systems: ACyber-Physical Systems Approach. Lee and Seshia (2011). http://LeeSeshia.org
3. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-oriented computing: a research roadmap. Int. J. Coop. Inf. Syst. **17**, 223–255 (2008)
4. Jian, H., et al.: Real-time service-oriented distributed governance. In: 6th World Congress on Services, Miami, pp. 479–484 (2010)
5. Wang, T., Niu, C., Cheng, L.: A two-phase context-sensitive service composition method with the workflow model in cyber-physical systems. In: 2014 IEEE 17th International Conference on Computational Science and Engineering (CSE), pp. 1475–1482 (2014)
6. Guinard, D., Trifa, V., Karnouskos, S., Spiess, P., Savio, D.: Interacting with the soa-based internet of things: discovery, query, selection, and on-demand provisioning of web services. IEEE Trans. Serv. Comput. **3**, 223–235 (2010)
7. Theorin, A., Ollinger, L., Johnsson, C.: Service-oriented process control with grafchart and the devices profile for web services. In: Borangiu, T., Thomas, A., Trentesaux, D. (eds.) Service Orientation in Holonic and Multi Agent. SCI, vol. 472, pp. 213–228. Springer, Heidelberg (2013)
8. Haber, A., Ringert, J.O., Rumpe, B.: Montiarc-architectural modeling of interactivedistributed and cyber-physical systems (2014). arXiv preprint arXiv:1409.6578
9. Peng, W., Yang, X., Shao, H.Z.: Cyber-physical system components composition analysis and formal verification based on service-oriented architecture. In: IEEE Ninth International Conference on e-Business Engineering (ICEBE), Hangzhou, pp. 327–332 (2012)

10. Ringert, J.O., Rumpe, B., Wortmann, A.: A requirements modeling language for the componentbehavior of cyber physical robotics systems (2014). arXiv preprint arXiv:1409.0394
11. Dorf, R.C., Bishop, R.H.: Modern Control Systems, 12th edn. Pearson, Boston (2011)
12. Derler, P., Lee, E.A., Vincentelli, A.S.: Modeling cyberphysical systems. Proc. IEEE **100**, 13–28 (2012)
13. Wikipedia. Hybrid system. http://en.wikipedia.org/wiki/Hybrid-system
14. Platzer, A.: Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics. Springer, Heidelberg (2010)
15. Platzer, A., Quesel, J.-D.: KeYmaera: a hybrid theorem prover for hybrid systems (system description). In: Armando, A., Baumgartner, P., Dowek, G. (eds.) IJCAR 2008. LNCS (LNAI), vol. 5195, pp. 171–178. Springer, Heidelberg (2008)
16. Qian, K.Y., Ye, L., Zhang, L.: A collaborative smart car system based on CPS services. In: Proceedings of National Conference on Services Computing 2015, Xianning, China (in Chinese)
17. Meredith, L.G., Bjorg, S.: Contracts and types. Commun. ACM **46**, 41–47 (2003)
18. Castagna, G., Gesbert, N., Padovani, L.: A theory of contracts for web services. In: ACM SIGPLAN Notices, vol. 43, pp. 261–272. ACM (2008)
19. Brogi, A.: On the potential advantages of exploiting behavioural information for contract-based service discovery and composition. J. Logic Algebraic Program. **80**, 3–12 (2011)
20. Uckelmann, D., Harrison, M., Michahelles, F.: Architecting the Internet of Things. Springer, Heidelberg (2011)
21. Hu, X., Chu, T.H., Chan, H.C., Leung, V.C.: Vita: a crowdsensing-oriented mobile cyber-physical system. IEEE Trans. Emerg. Top. Comput. **1**, 148–165 (2013)
22. Garcia Valls, M., Lopez, I.R., Villar, L.F.: iLAND: an enhanced middleware for real-time reconfiguration of service oriented distributed real-time systems. IEEE Trans. Ind. Inform. **9**, 228–236 (2013)
23. Thramboulidis, K.: An open distributed architecture for flexible hybridassembly systems: a model driven engineering approach (2014). arXiv preprint arXiv:411.1307
24. Ye, L., Tang, P., Guo, L.P., Zhang, L.: Modeling and verifying services of internet of things based on hybrid system methodology. J. Chin. Comput. Syst. **34**, 2663–2668 (2013). (in Chinese)
25. Tang, P., Ye, L., Guo, L.P., Zhang, L.: Composition and verifying of internet of things system. Comput. Eng. **39**, 45–48 (2013). (in Chinese)