Affordances in Representing the Behaviour of Event-Based Systems

Fahim T. IMAM [a,1], Thomas R. DEAN [b]
[a] *School of Computing, Queen's University, Canada*
[b] *Department of Electrical and Computer Engineering, Queen's University, Canada*

Abstract. In this paper, we examine affordances within the context of representing the functional behaviour of event-based systems. We provide an overview of the ontology that supports such representation. We provide an example use of the affordances that can facilitate us to describe, model, and reason about the behaviour of event-based systems in an intuitive, effective manner.

Keywords. ontologies, affordances, functional reasoning, artificial intelligence

1. Introduction

In this paper, we examine affordances within the context of representing the functional behavior of event-based systems. The representational facility that we are going to investigate is called the Event-Based Functional Behavior Ontology (EFBO) [1]. Our previous work, Imam et al. [2], was dedicated to providing the detailed understanding of the EFBO, along with a set of use case scenarios of the ontology. The key motivation of that effort was to develop an effective representational facility that could be utilized to validate the levels of functional consistencies among a set of cross-platform, event-based application systems. The focus of this short paper, however, is to discuss the affordances within the EFBO's representational facility. We will observe how the EFBO along with its affordances can facilitate us to model and reason about a system's behaviour using a set of simple, intuitive expressions. As will be observed, through the use of its affordances, the EFBO provides a useful mechanism to comprehend, model, and reason about a system's functional behaviour in an effective, intuitive manner.

Affordances and Functionality. Originated in the field of perceptual psychology, the term *affordance* refers to an action possibility formed by the relationship between an agent and its environment [3]. This latter concept of affordance by Gibson [3] was adapted into the field of AI by Sahin et al. [4] as a design methodology for developing the AI systems. Affordances within the AI world are typically used to derive the potential actions that can stem from a set of relationships between the properties of an autonomous agent and the properties of its environment. An affordance, in its original sense, does not exist as a direct property of an agent or its environment. Within the context of the EFBO,

[1] Corresponding Author: Fahim T. Imam, School of Computing, Queen's University, ON K7K 3Z5, Canada; E-mail: fahim.imam@queensu.ca

the notion of functionality forms the analogous types of relationships as the notion of affordances do for the AI systems. We define the functional behavior of a system as a set of relations F that can exist between the agents and the events of that system, where F must correspond to the notion of affordances. For our purpose, we consider the following notion of affordances by Stoffregen [5] as adapted from Nye et al. [6]:

$$W_{pq} = (X_p, Z_q) \text{ and } h = f(p,q); \text{ where, } h \notin X_p \text{ and } h \notin Z_q.$$

In this formula above, W_{pq} represents a system that includes the part of its environment X_p and an agent Z_q, where p and q respectively represents the properties of X_p and Z_q. The potential affordance h exists between p and q through the relationship f. This formulation suggests that the affordance h exists within the system W_{pq} where neither the environment X_p nor the agent Z_q contain the affordance h.

2. The EFBO Representation

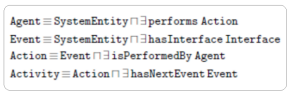
Based on the key notions of *action*, *events*, and *change* in commonsense reasoning [7], event calculus [8], and functional reasoning [9], the EFBO was developed to represent the functional behavior of a system in a rigorous, logical manner. The EFBO can be represented in terms of the following sets of classes and properties [2].

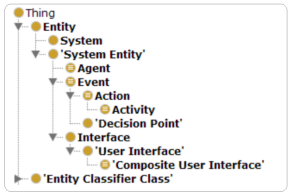
- Set of classes, $C = \{E, G, I\}$, where, E is the set of *Event* instances, E=$\{e_1, e_2, .., e_n\}$; G is the set of *Agent* instances, G=$\{g_1, g_2, .., g_n\}$; and, I represents the set of $Interface$ instances, I=$\{i_1, i_2, .., i_n\}$.
- Set of properties, P=$\{R(e,g), R(e,i), R(e_1,e_2)\}$, where, $R(e,g)$ represents the set of relations between an instance of an event, $Event(e)$ and an instance of an agent, $Agent(g)$, e.g., *isPerformedBy*; $R(e,i)$ represents the set of relations between an $Event(e)$ and an *Interface(i)*, e.g.,*hasInterface*; and, $R(e_1,e_2)$ represents the set of relations between two *Event* instances, e_1 and e_2; e.g., $hasNextEvent$.

The EFBO facilitates us to describe the instances of the classes in C in terms of their associated relations in P. The key classes and the classification of the main properties at the instances level are presented in Figure 1. The core classes of the EFBO are the *Event*, *Agent*, and the *Interface* classes. Other EFBO entities are logically based on these classes. The concept of *Event* within the EFBO essentially correspond to the notion of SPAN as endorsed by the Basic Formal Ontology (BFO) [10]. An event is understood as an occurrent at a particular point in time. The concept of time is understood as the continued progression of events that occur in succession from the past through the present to the future[2]. These latter notions of time and event together forms the Event-Time continuum as depicted in Figure 2 (right).

The temporal dimension of the events within the EFBO can be reasoned through the *Event-Event* relations as listed in Figure 1. The concept of *Agent* is understood as an actor that performs some action within a system environment. An *Interface* is understood as a system entity that serves as an interaction point between an event and an agent. We are only concerned about the temporal aspects of the interfaces through their relations with Events. Figure 2 (left) is a depiction of the *Event-Interface-Agent* interaction model as supported by the EFBO.

[2]We adapted the notion of time from the Oxford Dictionaries.





- Number of Classes: 18
- Number of Properties: 30
- Number of Axioms: 137
- DL-Expressivity: SROIQ(D)
- Formalizm Used: OWL 2
- Reasoner Used: HermiT 1.3

	Property	Description	Purpose
Event-Event	*hasNextEvent*	Non-transitive relation to connect an event to its immediate next event.	Assertion
	hasPreviousEvent	Inverse of the above – a non-transitive relation to connect an event to its immediate previous event.	Inference
	hasFutureEvent	Transitive relation between an event and its future event. This is the super property of the chain of *hasNextEvent* property; i.e., if an event *x* has a next event *y*, and *y* has a next event *z*, then *z* is the future event of *x*.	Inference
	hasPastEvent	Inverse of the above.	Inference
	isAlternateEventOf	Symmetric relation between two mutually exclusive events. This property, therefore, is specified as disjoint with the *hasParallelEvent* property.	Assertion
	hasParallelEvent	Symmetric relation between two events that occurs in parallel.	Inference
Event-Interface	*hasInterface*	Relation between an event and its associated interface. In EFBO, an event must have a dedicated interface.	Assertion/ Inference
	isInterfaceOf	Inverse of the above.	Assertion/ Inference
Agent-Interface	*interactsWith*	Symmetric relation between an agent and an interface. An agent should have no direct means to interact with a system's event without the event's dedicated interface.	Assertion/ Inference
Interface-Interface	*hasElement*	Transitive, partonomic relation between two interfaces. A user interface (UI), for example, can have multiple UI elements as its part. An Application Program Interface (API) can also be a composition of multiple APIs.	Assertion/ Inference
	isElementOf	Inverse of the above.	Assertion/ Inference

Figure 1. On the left, we note the key classes of the EFBO and their hierarchy, along with the current metrics of the ontology. On the right, we have the classification of the key EFBO properties. The classification is presented based on the four basic types of relations as supported by the EFBO. Some of the properties must have direct assertions of their instances whereas the other ones must infer their instances through automated reasoning as indicated in the last column. Essentially, the last five properties in the table can serve alternative purposes due to the existence of their corresponding inverse properties.

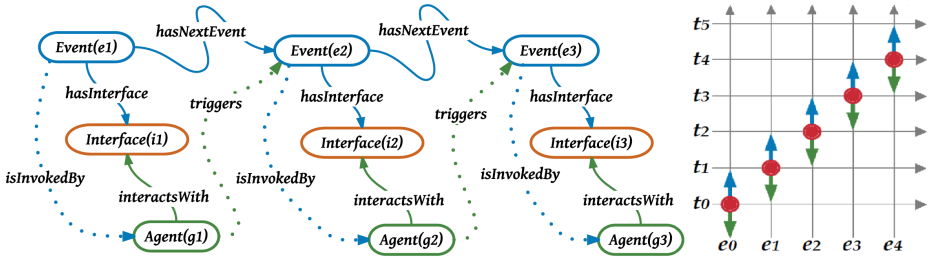


Figure 2. On the left, we have the *Event-Interface-Agent* interaction model for the EFBO. The *hasNextEvent* relation is used to connect an event with its next event. The corresponding interfaces for each of the events are connected through the *hasInterface* property. The *interactsWith* property is used to specify the agents for each of the interfaces. The dotted arrows indicate the inferred relations within the model. On the right, we have a depiction of the *Event-Time* continuum. The green and the blue arrows for each of the events at the red dots indicate the event's temporal connections with its past and future events, respectively. [2]

3. Affordances within the EFBO

Based on the formulation of affordances in Section 1, a common approach of implementing affordances is by defining the relationship f in terms of the properties of the agent q and the environment p [6]. The affordances within the EFBO are defined in terms of the relations between the properties of the *Agent* and the properties of the remaining system entities, i.e., the *Event* and the *Interface*. The following affordances are defined using the powerful notion of the OWL-property chain.

A. The *isPerformedBy* Relation. The concept of *Action* within the EFBO is specified as an event that is performed by an agent. However, an action cannot be performed directly without invoking the interface of its corresponding event. We define the relational property *isPerformedBy* as the super property of the following chain of properties:

hasInterface* o *interactsWith, where, the *hasInterface* is an *eventProperty* to connect an event with an interface; and,the *interactsWith* is an *agentProperty* that connects an agent with an interface. The property chain specifies that the affordance *isPerformedBy* could exist if an $Event(e)$ has an *Interface(i)* and that interface *i* interacts with an $Agent(g)$. The event e, in this case, would be inferred as an action that is performed by the agent g. The ***isInvokedBy*** relation refers to the super property of the defined relation *isPerformedBy*. The former is a relation between an event and agent, and the latter is between an action and an agent. The instances that hold these two properties can only be inferred and must not be asserted directly. These relations, therefore, can only be afforded by the chain of relations between the *hasInterface* and the *interactsWith* properties.

B. The *isTriggeredBy* Relation. The concept of *Activity* within the EFBO is understood to be an *Action* (i.e., an event performed by an agent) that is followed by a successive event. In order to activate an event, the event must be triggered by an agent through the system's interface. We define the *isTriggeredBy* relation as the super property of the following chain of properties: ***hasPreviousEvent* o *isPerformedBy***, where, the *hasPreviousEvent* is an *eventProperty* that connects an event with its previous event; and, *isPerformedBy* is a defined property that connects an action with an agent as explained before. The property chain specifies that the affordance *isTriggeredby* can exist if an $Event(e_2)$ has a previous $Event(e_1)$, and the event e_1 is performed by an $Agent(g)$. The event e_2, in this case, would be inferred as an event that is triggered by the agent g. The instances that hold the *isTriggeredBY* properties can only be inferred and must not be asserted directly. In other words, the *isTriggeredBy* can only be afforded by the chain of relation between the *hasPreviousEvent* and the *isPerformedBy* properties.

As we can observe, the affordances within the EFBO facilitates us to have a straightforward mechanism to specify different relationships between the properties of an agent and its system environment. We can also observe that the EFBO allows nesting of affordances, as well as defining affordances in terms of the other affordances. This can be a powerful mechanism in order to have different levels of abstractions of the potential actions by an agent within the event-based systems.

Example Use of the EFBO's Affordances. As an example, we observe the EFBO-based modelling of a typical login functionality for a smart phone application [2]. While the domain of the example is simple enough to follow, a careful observation of the example should intuitively indicate most of the necessary aspects of modelling a more complex system. The EFBO-based modelling process involves describing the behavior of a system on a simple storyboard, as reflected in Figure 3. As should be observed in the figure, the storyboard allowed us to state whatever is *explicitly known* about each of the core functional entities, i.e., the associated relations between the events, relations between the events and their interfaces, and the relations between the agents and the interfaces of the desired system.

Based on a set of functional reasoning categories, e.g., events by triggering agents, distribution of activities by agents, events within a specific activity, and so forth, a set of instance-specific classes (the class names prefixed with underscores in Figure 4) were defined under the 'Entity Classifier Class' of the EFBO. For example, the class '*_User Agent Activity*' was defined as equivalent to any *Action* that *isPerformedBy* some *_userAgent*. Similarly, the class '*_Event Triggered By Client Agent*' was defined as equivalent to any *Event* that *isTriggeredBy* some *_clientAgent*. Figure 4 presents the inferred classification results after the automated reasoning on the classifier classes.

```
1  //AN EXAMPLE OF A STORYBOARD FOR A LOGIN FUNCTIONALITY OF A MOBILE APPLICATION.
2
3  //DECLARATIONS OF THE SEQUENCE OF EVENTS.
4  EVENT_START hasNextEvent _tapAppIcon
5  _tapAppIcon hasNextEvent _launchAppInterface
6    _launchAppInterface hasNextEvent _enterUserInfo
7          _enterUserInfo hasNextEvent _enterUserName
8          _enterUserName hasNextEvent _enterPassword
9          _enterPassword hasNextEvent _tapLoginButton
10   _tapLoginButton hasNextEvent _sendUserInfo
11         _sendUserInfo hasNextEvent _verifyUserInfo
12               _verifyUserInfo hasNextEvent _presentTryAgainUI
13                     _presentTryAgainUI hasNextEvent _enterUserInfo
14                     _presentTryAgainUI isAlternateEventOf _presentWelcomeUserUI
15               _verifyUserInfo hasNextEvent _presentWelcomeUserUI
16                     _presentWelcomeUserUI hasNextEvent EVENT_END
17
18 //DECLARATIONS OF INTERFACES FOR EACH OF THE EVENTS.
19 _tapAppIcon hasInterface _appIcon
20 _launchAppInterface hasInterface _appInterface
21 _enterUserInfo hasInterface _loginInterfaceUI
22         _loginInterfaceUI hasElement _userNameField
23               _userNameField isInterfaceOf _enterUserName
24         _loginInterfaceUI hasElement _passwordField
25               _passwordField isInterfaceOf _enterPassword
26         _loginInterfaceUI hasElement _loginButton
27               _loginButton isInterfaceOf _tapLoginButton
28
29 //CONTINUED: DECLARATIONS OF INTERFACES FOR
30 //EACH OF THE EVENTS.
31 _sendUserInfo hasInterface _clientInterface
32 _verifyUserInfo hasInterface _serverInterface
33 _presentWelcomeUserUI hasInterface _welcomeUserUI
34       _welcomeUserUI hasElement _welcomeLabel
35 _presentTryAgainUI hasInterface _tryAgainUI
36       _tryAgainUI hasElement _tryAgainLabel
37
38 //DECLARATIONS OF THE AGENTS INTERACTIONS.
39 _userAgent interactsWith _appIcon
40 _userAgent interactsWith _loginInterfaceUI
41 _userAgent interactsWith _userNameField
42 _userAgent interactsWith _passwordField
43 _userAgent interactsWith _loginButton
44 _serverAgent interactsWith _serverInterface
45 _clientAgent interactsWith _appInterface
46 _clientAgent interactsWith _welcomeUserUI
47 _clientAgent interactsWith _tryAgainUI
48 _clientAgent interactsWith _clientInterface
49 //END OF THE STORYBOARD.
```

Figure 3. The EFBO-based modelling statements on a storyboard. First, the sequence of events of the login system are declared using the *hasNextEvent* property (Lines 4-16). After that, the set of interfaces for each of the events are declared using the *hasInterface* property; the composition of the UI interfaces are specified using the *hasElement* / *isElementOf* properties (Lines 19-36). Finally, the set of agents with their designated interfaces are declared using the *interactsWith* property (Lines 39-48). [2]

Figure 4. A set of inferred classifications of the Figure 3 entities after automated reasoning. [2]

It should be observed from the modelling example that the EFBO's affordance relations allowed the reasoner to infer the set of potential actions between the system's agents and its environment. For example, the reasoner perfectly inferred the actions that could be performed by different agents as part of their activities within the modelled system; the reasoner could also infer the events that could be triggered by each of the agents as well (see the first two columns in Figure 4). It should be noted that while we did not have the EFBO affordances directly asserted between any of the storyboard entities, the affordance relations were *inferred* through the automated reasoning as afforded by the relations between the properties of the events and the properties of the agents within the modelled environment. This latter strategy is critical in both observing and modelling the functional behavior of any practical application system. In the practical

sense of affordances, the agents in such system must not be able to invoke the functional events of the system without the proper interactions with the system's designated interfaces. This design choice also promotes more automation, and, ultimately, reduces the amount of human-induced errors during the modelling process. As observed in this section, the relations of affordances can be a powerful feature for the ontologies that deals with representing the functional behaviour of event-based systems.

4. Concluding Remarks

Using the EBFO representation along with its affordances, one can effectively model the functional entities of a system in such a way so that their existence within the system can be thoroughly reasoned. It should be noted that currently, the temporal aspects of the EFBO events only supports a subset of Allen's temporal axioms that are necessary for our functional reasoning tasks. However, the EFBO is flexible enough to incorporate the remaining axioms, if necessary in the future. Due to its affordances, the EFBO-based functional modelling process is quite intuitive and practical, which allows us to describe and reason about the behavior of a system through a series of simple statements. We have carefully engineered the EFBO ontology with a minimal set of required classes and properties in order to achieve the maximum usability of the ontology without compromising the required logical rigour to achieve an effective functional reasoning mechanism.

Acknowledgement

This work is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] F. T. Imam. The Event-Based Functional Behavior Ontology (EFBO). Queen's University, Canada. `http://cs.queensu.ca/~imam/efbo.html`. [Last accessed: 15-June-2016].

[2] F. T. Imam and T. R. Dean. Modelling functional behavior of event-based systems: A practical knowledge-based approach. In *Proc. of the 20th Int'l Conf. on Knowledge Based and Intelligent Information and Engineering Systems, York, UK*. Procedia Computer Science, Elsevier, 2016. In Press.

[3] J. J. Gibson. The ecological approach to the visual perception of pictures. *Leonardo*, 11(3):227–235, 1978.

[4] E. Şahin, M. Cakmak, M. R. Doğar, E. Uğur, and G. Üçoluk. To afford or not to afford: A new formalization of affordances toward affordance-based robot control. *Adaptive Behavior*, 15(4):447–472, 2007.

[5] T. A. Stoffregen. Affordances as properties of the animal-environment system. *Ecological Psychology*, 15(2):115–134, 2003.

[6] B. D. Nye and B. G. Silverman. Affordances in AI. In *Encyclopedia of the Sciences of Learning*, pages 183–187. Springer, 2012.

[7] E. Davis and G. Marcus. Commonsense reasoning and commonsense knowledge in artificial intelligence. *Communications of the ACM*, 58(9):92–103, 2015.

[8] M. Shanahan. The event calculus explained. In *Artif. intell. today*, pages 409–430. Springer, 1999.

[9] B. H. Far and A. H. Elamy. Functional reasoning theories: Problems and perspectives. *AIE EDAM*, 19(02):75–88, 2005.

[10] P. Grenon and B. Smith. SNAP and SPAN: Towards dynamic spatial ontology. *Spatial cognition and computation*, 4(1):69–104, 2004.