Ignorance is Bliss: A Complexity Perspective on Adapting Reactive Architectures

Todd Wareham
Department of Computer Science
Memorial University of Newfoundland
St. John's, NL Canada A1B 3X5
Email: harold@mun.ca

Johan Kwisthout
Institute for Computing and
Information Sciences,
Radboud University
Nijmegen, the Netherlands
Email: j.kwisthout@science.ru.nl

Pim Haselager and Iris van Rooij
Donders Institute for
Brain Cognition and Behaviour,
Radboud University
Nijmegen, the Netherlands
Email: w.haselager@donders.ru.nl,
i.vanrooij@donders.ru.nl

***Abstract*—We study the computational complexity of adapting a reactive architecture to meet task constraints. This computational problem has application in a wide variety of fields, including cognitive robotics, evolutionary robotics and cognitive neuroscience. We present a proof that—even for a rather simple world and a simple task—adapting a reactive architecture to perform a given task in the given world is NP-hard. The result implies that adapting reactive architectures is computationally intractable regardless the nature of the adaptation process (*e.g.*, engineering, development, evolution, learning, etc.) unless very special conditions apply. In order to find such special conditions for tractability, we have performed parameterized complexity analyses. One of our main findings is that architectures with limited sensory and perceptual abilities are efficiently adaptable.**

I. Introduction

A popular class of control architectures in behavior-based robotics are the hybrid deliberative/reactive architectures. These architectures combine the flexibility of a high-level deliberative system incorporating planning, world-knowledge, and memory with the speed and robustness of a low-level reactive system [1], [2]. A key issue for such architectures is linking of these two components. One approach to this is for the deliberative component to adapt the reactive component in response to changing conditions, either by reconfiguring the interactions of existing reactive behaviors or augmenting existing behaviors with newly-designed ones [1, p. 214].

There are a number of robotic implementations which show that reactive adaptation is possible, *e.g.*, Autonomous Robot Architecture (AuRA) (see [1, Section 6.6.1] and references), Planner-Reactor [3], SSS [4]. These implementations show that one can efficiently adapt reactive architectures to meet *certain* task constraints in *certain* situations. An open question is to what extent such implementations can generalize to, and scale for, other types of tasks and situations.

Computational complexity theory provides techniques for addressing this question, and some preliminary results are known. For instance, Selman [5] found that devising a reactive plan for solving a given planning task is NP-hard (see also [6]), and Dunne, Laurence, and Wooldridge [7] proved that designing reactive agents that can perform a given achievement or maintenance task is NP-hard (see also [8], [9], [10]). Yet, the formalisms adopted by these researchers to model the reactive architectures and their life worlds were of such a high degree of generality that the intractability results may be due more to the formalisms used than the complexity inherent in adaptation. As our interest here is primarily in the latter, we study specifically the complexity of adapting subsumption reactive architectures [11] relative to simple static worlds.

Using techniques from classical complexity theory [12], we show that adapting a subsumption-based reactive architecture so that it can navigate a given world is NP-hard. This holds true, regardless whether the adaptation occurs by reconfiguring reactive-behavior layers in the architecture or by designing reactive-behavior layers anew and adding them to the architecture. These results indicate that adapting reactive architectures is computationally intractable unless very special conditions apply. This raises the question of which conditions characterize those situations in which adapting reactive architectures is tractable. An answer to this question may be relevant for various approaches to robotics. For instance, it can inform roboticists about the conditions that make adaptation of reactive architectures—*e.g.*, as done by deliberative modules in hybrid architectures or by an evolutionary algorithm for adapting a reactive robot to a task environment—feasible. Moreover, it is of interest to cognitive neuroscience as it can inspire hypotheses about evolutionary or developmental explanations of animal and human brain structure.

In order to find conditions for tractability, we performed parameterized complexity analyses [13] of the problem of adapting reactive architectures. Our analyses reveal that restrictions on either the internal structure of the architecture or the perceptual complexity of its sensory inputs render adaptation tractable. Though these results are derived in the context of a specific navigation task, we will show that they apply to any task for which a given candidate architecture can be verified efficiently in a given world.

The remainder of this paper is organized as follows. In Section II, we formalize reactive adaptation in terms of a simplified subsumption architecture for a basic navigation task, distinguishing between two forms of adaptation: reconfiguration and design. Section III gives proofs of the general intractability of both of these problems. Section IV describes a methodology for identifying conditions for tractability, which is then applied in Section V to identify such conditions for the reconfiguration and design variants of adapting reactive architectures. Finally, our conclusions and directions for future work are given in Section VI.

II. Formalizing Reactive Adaptation

A. Adaptation for a Task

An adaptation mechanism for a particular robot architecture A that will enable A to perform some task T relative to a world W can be construed as a mechanism that adapts A in some limited fashion so that it can perform T in W. The computation performed by this mechanism is modeled by the following informal computational problem:

T-Adaptation by M (TA-M)
Input: World W, an architecture that can only partially perform task T in W, and an integer d.
Output: An architecture A' derived from A by at most d modifications of type M that can fully perform T in W, if such an A' exists, and special symbol $\perp$ otherwise.

Analyses of the computational complexity of this problem can show both (1) whether any adaptation mechanisms suffices, *i.e.*, can operate in a reasonable amount of time and space, relative to the particular choices of world, architecture, task, and architecture-modifications, and if not, (2) under which combinations of restrictions on these choices such adaptation might be possible.

B. Adaptation for Basic Navigation

In this section, we will consider a particular formalization of problem TA-M relative to the task of basic navigation for a simplified subsumption-based reactive architecture.

Our worlds of interest will be finite square-based maps in which compass movement is possible between adjacent squares, *i.e.*, north, south, east, and west, and each square is either a freespace (which a robot can occupy or travel through) or an obstacle. Each square has an associated type that is recognizable by the sensors on the robot; let this set of types be denoted by $|E|$ and the number of such types by E.

Within such a world, the navigation task will be to, starting from an arbitrary initial freespace, move to eventually occupy another arbitrary final freespace denoted by a specially-marked square-type. A robot that can do this relative to any two initial and final freespaces in W is said to be fully navigable for W; otherwise, the robot is partially navigable for W. Note there will be no optimality restrictions on the paths eventually navigated, *e.g.*, the path travelled need not be the shortest possible between the initial and final freespaces.

Our robot will be a simplified subsumption-based reactive architecture consisting of sensors, a set of layers, a total ordering on these layers, and a set of subsumption connections between layers. The sensors can see outwards in a radius r around the robot in every direction up to the closest obstacle in that direction, and can only verify, for each square-type $e \in E$, the presence of e within that perceptual radius, *i.e.*, $exists(e)$. Each layer will consists of a single trigger-condition that is a Boolean formula of length f over the available sensory $exists$-predicates and an action $a \in \{N, S, E, W\}$. If a layer's formula evaluates to $True$, the layer produces output a; otherwise, it produces the special output null. Given a set of layers L, we will assume that the formula in each layer contains at least one $exists$-predicate and no two layers encode formulas that both compute the same Boolean function and produce the same output. Relative to the total order on the layers, a layer i can have subsumption-links to any layer j that is lower than i in the ordering; between any two layers, there can exist an output-inhibition or output-override link (but not both). The output of any layer that subsumes at least one lower-level layer is not available directly for output; otherwise, that layer's output is available. The output of a set of ordered layers with subsumption links will be that of the highest layer relative to the order that is both available and non-null.

Relative to such a reactive architecture, we will consider two types of architecture modifications for adaptation:

1) Adding a selection of layers from a specified layer-library, along with some number of subsumption-link additions and deletions; and
2) Adding a selection of possible layers, along with some number of subsumption-link additions and deletions.

These modifications correspond to those considered in [1], [3], [4]. In combination with the above, these modes of modification yield the following formalizations of problem TA-M relative to the navigation task and subsumption-based reactive architectures:

Navigation Adaptation by Reconfiguration
Input: A world W, a subsumption architecture A that is only partially navigable for W, a library M of layers, and integers s and l.
Output: A subsumption architecture A' derived from A by the addition of at most l layers from M and the addition or deletion of at most s subsumption-links that is fully navigable for W, if such an A' exists, and special symbol $\perp$ otherwise.

Navigation Adaptation by Design
Input: A world W, a subsumption architecture A that is only partially navigable for W, and integers s and l.
Output: A subsumption architecture A' derived from A by the addition of at most l layers and the addition or deletion of at most s subsumption-links that is fully navigable for W, if such an A' exists, and special symbol $\perp$ otherwise.

These problems will be denoted below by NA-REC and NA-DES, respectively. Note that layers may be added in any order relative to the layers in A in both problems, and that all layers have trigger-formulas of length $\leq f$.

III. Reactive Adaptation is Intractable

In this section, we address whether or not reactive adaption for the navigation task can be done efficiently relative to the subsumption architecture and architecture modifications described in Section II. Following general practice in Computer

Science, we define efficient solvability as being solvable in the worst case in time polynomially bounded in the input size, and show that a problem is not polynomial-time solvable by proving it to be at least as difficult as the hardest problem in problem-class NP, *i.e.*, NP-hard (see [12], [14] for details). Our proofs will involve polynomial-time reductions from the following well-known NP-hard problem:

DOMINATING SET
Input: A graph $G = (V, A)$ and an integer k.
Question: Is there a dominating set in G of size at most k, *i.e.*, is there a subset $V' \subseteq V$, $|V'| \leq k$, such that for each $v \in V$, either $v \in V'$ or $\exists (v, v') \in E$ such that $v' \in V'$?

Theorem 1: NA-REC is NP-hard

Proof: Given an instance $I = \langle G, k \rangle$ of DOMINATING SET, construct the following instance $I' = \langle W, A, M, s, l \rangle$ of NA-REC: Let $E = \{U, D, L, A, B, \%, V_1, V_2, \ldots V_{|V|}, -, F\}$, with F being freespace and all other squares being obstacles. World W is a ring-shaped track of freespaces surrounded by obstacles on both the inner and outer sides of the track. This track can be divided into north, east, south, and west regions, and each region has inner and outer sides. The track is specified as follows:

- the east region consists of D-squares on both sides.
- the south region consists of L-squares on both sides.
- the west region consists of U-squares on both sides.
- The north region has an initial pair of A-squares on the left and a final pair of B-squares on the right. In between are $|V|$ blocks of length $|V|$ if $|V|$ is odd and $|V| + 1$ otherwise apiece separated by pairs of A-squares. Given an arbitrary order on the vertices in V, square i in the top side of each block corresponds to vertex i under the ordering; when $|V|$ is even, the vertices in V are split into two even-length sub-blocks with a middle spacer-square. Block i, $1 \leq i \leq |V|$, corresponds to a particular vertex $v_i \in V$, and encodes the vertices adjacent to v_i (including v_i itself) on the top side of the block (with vertex-positions not in the neighbourhood (or the middle spacer-square, if $|V|$ is even) marked with $-$-squares) and all $\%$-squares on the bottom side of the block.

A graph G and its associated world are shown in Fig.1. Note that regions are modified at corners of the track to ensure that an architecture can change direction at the corners under the layer-ordering and visibility constraints below. Architecture A has perception-radius $r = \lfloor |V|/2 \rfloor$ and consists of $|V| + 6$ layers, such that the layers in descending order are:

- A layer that issues a E action if an A-square is detected;
- A layer that issues a N action if a $\%$-square is detected;
- $|V|$ layers, one for each vertex $v \in V$, that issue E actions if the square corresponding to v is detected;
- A layer that issues a N action if a U-square is detected;
- A layer that issues a W action if a L-square is detected;
- A layer that issues a S action if a D-square is detected; and
- A layer that issues a E action if a B-square is detected.

The subsumption links are output-inhibition links from the topmost layer to each of the vertex layers. Such an architecture for an example graph G and associated world W is shown in part (c) of Fig.1. Finally, let $M = \emptyset$, $s = k$, and $l = 0$. Observe that this construction can be done in time polynomial in the size of the given instance I of DOMINATING SET.

The following observations will be useful:

1) Both A and any A' created by subsumption-link modification can only move in a clockwise fashion in W.
2) A can move past any freespace in W except the middle freespace in each vertex block. This is so because in each such middle freespace, neither of the A or B-squares surrounding the vertex-block which would allow A to move forward can be sensed (as $r = \lfloor |V|/2 \rfloor$).
3) The only way any A' created from A by subsumption-link modification can progress past the middle freespace in a vertex block is to remove the inhibition links from the topmost layer to one or more of the vertex layers in A corresponding to a vertex-square that is present in that block (which could be sensed under the given r).

The above implies that to make any A' derived from A by subsumption-link modification fully navigable for W, those link modifications must enable A' to progress past the middle freespace of each vertex-block in W.

To prove that this construction is a reduction, we must show that the answer to the given instance of DOMINATING SET is "Yes" if and only if the constructed instance of NA-REC has an associated A' that is fully navigable for W. Let us consider the two implications separately. If the answer to the given instance of DOMINATING SET is "Yes", then there is a dominating set V' such that $V' \leq k$. Construct A' from A by deleting the subsumption links from the topmost layer to all $|V'| \leq k = s$ vertex layers corresponding to vertices in V'. These layers will now be active whenever they detect squares corresponding to vertices in V' in the vertex blocks in W, which, by the observations above, will allow A' to navigate between any two freespaces in W. Conversely, if there is an A' that is fully navigable for W, A' was constructed from A by removing at most s subsumption links from A; let V' be the set of vertices in G corresponding to the now-active vertex layers in A'. By the construction of W, each vertex $v \in V$ is either in V' or is adjacent to a vertex in V', implying that V' is a dominating set of size at most $s = k$ in G. ■

Theorem 2: NA-DES is NP-hard

Proof (sketch): Follows by a slight variant of the reduction in Theorem 1 which deletes M and explicitly sets $f = 1$. The proof of correctness of this reduction is identical to that for Theorem 1. ■

Modulo the conjecture $P \neq NP$ which is widely accepted to be true [14], the above shows that neither NA-REC nor NA-DES are polynomial-time solvable. Moreover, as $|M| = l = 0$ in both proofs, this holds even if *no* new layers are added to the architecture, *i.e.*, the only allowable modifications are to the subsumption-links between existing layers.

```
        (a)

      1   4
      |\  |
      | \ |
      |  \|
      2---3

        (b)

UUUA12-3-A12-3-A12-34A--34BBBB
U............................B
U.UA%%%%%A%%%%%A%%%%%A%%%%BB.B
U.U                        D.D
L.LLLLLLLLLLLLLLLLLLLLLLLLLD.D
L............................D
LLLLLLLLLLLLLLLLLLLLLLLLLLDDDD
```

```
         (c)

e(A)? -> E ------>
e(%)? -> N--++++
             ||||
e(1)? -> E -i---->
              |||
e(2)? -> E --i--->
               ||
e(3)? -> E -i-i-->
                |
e(4)? -> E ----i->
e(U)? -> N ------>
e(L)? -> W ------>
e(D)? -> S ------>
e(B)? -> E ------>
```

Fig. 1. Illustration of reduction from DOMINATING SET to NA-REC given in Theorem 1. a) Sample graph G. b) World W constructed from G. c) Subsumption architecture A associated with G and W. For clarity, freespace squares are denoted by a period instead of F in (b).

IV. A METHOD FOR IDENTIFYING TRACTABILITY CONDITIONS

A computational problem that is intractable for unrestricted inputs may yet be tractable for non-trivial restrictions on the input. This insight is based on the observation that some NP-hard problems can be solved by algorithms whose running time is polynomial in the overall input size and non-polynomial only in some aspects of the input called *parameters*. In other words, the main part of the input contributes to the overall complexity in a "good" way, whereas only the parameters contribute to the overall complexity in a "bad" way. In such cases, the problem Π is said to be **fixed-parameter tractable** for that respective set of parameters. The following definition states this idea more formally.

Definition 1: Let Π be a problem with parameters k_1, k_2, ... Then Π is said to be *fixed-parameter tractable* for parameter-set $K = \{k_1, k_2, ...\}$ if there exists at least one algorithm that solves Π for any input of size n in time $f(k_1, k_2, ...)n^c$, where $f(\cdot)$ is an arbitrary computable function and c is a constant. If no such algorithm exists then Π is said to be *fixed-parameter intractable* for parameter-set K.

In other words, a problem Π is fp-tractable for a parameter-set K if all superpolynomial-time complexity inherent in computing the function can be confined to the parameters in K. In this sense the "unbounded" nature of parameters in K can be seen as a reason for the intractability of the unconstrained version of Π.

There are many techniques for designing fp-tractable algorithms [15], [16], and fp-intractability is established in a manner analogous to classical polynomial-time intractability using parameterized reductions (in which parameters are functions of each other) to show that a parameterized problem is at least as hard as the hardest problems in one of the problem-classes in the W-hierarchy $\{W[1], W[2], ...\}$ (see [13] for details).

Observe that it follows from the definition of fp-tractability that if an intractable problem Π is fp-tractable for parameter-set K, then Π can be efficiently solved even for large inputs, provided only that all the parameters in K are relatively small. This strategy for rendering (otherwise intractable) problem tractable has been successfully applied in a variety of areas (see [13], [17] and references). In the next section we report on our investigation of whether or not the same strategy may be used to render the problems NA-REC and NA-DES tractable.

V. WHAT MAKES REACTIVE ADAPTATION TRACTABLE?

There are a number of parameters in our conception of reactive adaptation that are both plausibly of small value in practice and whose restriction might hence render reactive adaptation tractable. Table I lists the parameters considered here, which can be broken into three groups:

1) Restrictions on the (perceived) world ($|E|$);
2) Restrictions on subsumption architectures ($|L|, f$); and
3) Restrictions on architecture-modification ($s, |M|, l$).

In the remainder of this section, we will assess the fp-tractability of NA-REC (Section V-A) and NA-DES (Section V-B) relative to these parameters, note how these results apply in more general settings (Section V-C), and discuss the implications of these results (Section V-D).

A. Results for Adaptation by Reconfiguration

Corollary 1: NA-REC is fp-intractable for parameter-set $\{s, f, l, |M|\}$.

Proof: Follows from the $W[2]$-hardness of DOMINATING SET for parameter-set $\{k\}$ and the reduction in Theorem 1, in which $s = k$, $f = 1$, and $|M| = l = 0$. ■

TABLE I
PARAMETERS CONSIDERED IN ANALYSES OF NA-REC AND NA-DES.

Param.	Definition	Appl.
$\|E\|$	Number of distinguishable square-types in world	All
$\|L\|$	Number of layers in derived architecture A'	All
f	Maximum length of layer trigger-formula	All
s	Maximum number of subsumption-link changes	All
l	Number of layers added to A	All
$\|M\|$	Number of layers in provided library	NA-REC

Theorem 3: NA-REC is fp-intractable for parameter-set $\{s, f, l, |L|\}$.

Proof (sketch): Modify the reduction in Theorem 1 to let W be such that the bottom-side squares in the vertex blocks are now the same as the top-side squares, *i.e.*, eliminate the %-squares, A consist of the lowest five layers and the topmost layer of the original A, M consist of the vertex-layers from the original A, $s = 0$, and $l = k$. The result then follows from the $W[2]$-hardness of DOMINATING SET for parameter-set $\{k\}$ and the reduction above from DOMINATING SET to NA-REC in which $s = 0$, $f = 1$, $l = k$, and $|L| \leq k + 6$. ■

Lemma 1: Given an architecture A, a world W, and initial and final positions s and d with W, whether or not A can navigate from s to d can be computed in $O(|W||A|)$ time.

Proof: The action computed by A at a position p in W can be determined in $O(|A|)$ time. Observe that the behavior of A at p is fixed, in that regardless of whether p has been encountered once or more by A, the same action is generated. Hence, when started at s, if A doubles back on any previously-entered square, A can never encounter d. As A can travel to at most $|W| - 2$ different squares before entering d. it can be determined in $|W|$ moves whether A can reach d from s. ■

Theorem 4: NA-REC is fp-tractable for parameter-set $\{|L|, |M|\}$.

Proof (sketch): The number of subsumption link-configurations of A is a function of $|L|$, *i.e.*, $3^{|L|(|L|-1)/2}$. As $s \leq |L|$ and $l \leq |M|$, the number of possible A' that can be generated from A is a function of $|L|$ and $|M|$. To complete the proof, observe that each such configuration A' of A can be evaluated wrt W in polynomial time (as there are $|W|(|W| - 1)/2$ source-destination pairs in W and the reachability for each pair can be determined in $O(|A||W|)$ time by Lemma 1) and checked to see if A and A' differ by at most s subsumption-links in $O(max(|A|, |A'|))$ time. ■

Theorem 5: NA-REC is fp-tractable for parameter-set $\{|E|\}$.

Proof (sketch): Both $|L|$ and $|M|$ are bounded by the number of possible Boolean functions over $|E|$ times the number of possible actions a layer may generate, *i.e.*, $|L| \leq 2^{2^{|E|}} \times 4$. Substituting this $|E|$-expression for each occurrence of $|L|$ and $|M|$ in the runtime of the algorithm described in Theorem 4 gives an algorithm for NA-REC that is fp-tractable for $\{|E|\}$. ■

B. Results for Adaptation by Design

Corollary 2: NA-DES is fp-intractable for parameter-set $\{s, f, l, |L|\}$.

Proof (sketch): Modify the reduction in Theorem 3 to eliminate M and explicitly set $f = 1$. The result then follows from the $W[2]$-hardness of DOMINATING SET for parameter-set $\{k\}$ and the reduction above from DOMINATING SET to NA-DES in which $s = 0$, $f = 1$, $l = 0$, and $|L| \leq k + 6$. ■

Theorem 6: NA-DES is fp-tractable for parameter-set $\{|E|, f\}$.

Proof (sketch): Consider the following algorithm: By comparison against all possible values of the $exists$-predicates for E, possible layers over E that are not already part of A can be isolated; call this set of layers P. This set P can be further reduced to those layers that have trigger-condition formulas of length at most f over E; call this set P'. Evaluate all A' created by adding at most l layers from P' to and modifying at most s subsumption-links of A to see if any are fully navigable for W and output accordingly. To complete the proof, note that the non-polynomial quantities in the runtime of this algorithm (*i.e.*, the number of possible layers over E ($2^{2^{|E|}} \times 4$), the number of possible values of the $exists$-predicates for E ($2^{|E|}$), the number of trigger-condition formulas of length at most f ($\leq f \times |E|^f$)) are all functions of $|E|$ and f. ■

C. Generality of Results

As all fp-tractability results presented above require only that a candidate architecture be verifiable for a task relative to a given world in time polynomial in the size of that architecture and world (Lemma 1), these results hold relative to reactive adaptation for *any* such polynomial-time verifiable task. If in addition tasks in a given world are encoded as logical specifications and included in the input (as in [18]) and the basic navigation task and worlds used in our proofs can be encoded succinctly under the logics used, all NP-hardness and fp-intractability results hold as well.

D. Discussion

We have found that adapting reactive architectures, whether by recruiting pre-existing layers or by designing layers anew, is NP-hard (Theorems 1 and 2). This NP-hardness holds even for a basic navigation task in a simple 2D static world. Moreover, adapting reactive architectures remains NP-hard even if the adaptation is restricted to rewiring the given subsumption architecture, *i.e.*, without adding any new layers to it. These intractability results underscore the computational difficulty of adapting reactive architecture, be it by a human designer, a deliberative component in a hybrid robot, or by evolution, development or learning in a (human) brain.

To our knowledge, no explicit conjectures about the sources of computational difficulty in reactive adaptation have been made in the literature, but on the basis of successful reactive system design done by humans (consisting of small (≤ 10) numbers of layers developed in an incremental add-and-test manner ([19], [20]; see also [1, pp. 74–77])), it seems reasonable to conjecture that restrictions on the subsumption architectures ($|L|, f$) and degree of allowed modification ($s, l, |M|$) should render reactive adaptation tractable. However, it does not (Theorem 3 and Corollaries 1 and 2). What does result in tractability is when the total number of layers that can be used to configure a reactive architecture is small (*i.e.*, both $|L|$ and $|M|$ are small) (Theorem 4). Though useful to know, we can imagine this condition may be of limited interest or applicability for roboticists, as such reactive architectures will—by definition—have quite restricted behavioral repertoires.

Of greater interest, perhaps, is the second class of conditions for tractability that we have identified; viz., restrictions pertaining to the sensory and perceptual abilities of the architectures. For instance, we found that reconfiguring a reactive architecture to perform a task can be done efficiently provided only that the sensory sensitivity of the architecture (*i.e.*, the number of environmental features it can distinguish, $|E|$) is not too large (Theorem 5). In the more general case, where also newly designed layers can be added to the architecture, a simultaneous restriction of this sensory complexity *and* perceptual complexity—in the sense of the ability of layers to encode patterns in detected features ($|E|, f$)—renders adaptation of a reactive architecture tractable (Theorem 6).

Of course, these tractability results are modulo the assumption that for the task, world and architecture under consideration there *exists* a reactive architecture that can be constructed through adaptation and perform the task in the given world. One possible reading of our finding, then, is that adapting reactive control is feasible in environments that are structured such that the features and patterns that are relevant for successful behavior can be succinctly represented by a small set features and percepts of low complexity (cf. what [21] called "being ignorantly successful"). Although it is plausible that low perceptual complexity may characterized perception for humans [22], [23], probably only simpler organisms are characterized by low sensory sensitivity. It is expected then that if such simple organisms—or more generally, agents—enter an environment to which they can in principle adapt that they can do so quickly. Quick adaptation of more sensory-complex agents may still be possible, *e.g.*, by exploiting restriction on the classes of sensory information that can be detected (cf. the sensory modalities) or by exploiting a limited sensory radius (r), but if and how this could be done is an open question for future research.

VI. Conclusions

We have presented two formal characterizations of the problem of adapting a reactive architecture, one for reconfiguring such an architecture and one for designing parts of it anew. Our complexity analyses reveal that, while these problems are computationally intractable in general, there are conditions that render them tractable. Knowledge of these conditions can be exploited both in robotics design and in cognitive neuroscience to understand which properties a reactive architecture needs to have to be efficiently adaptable.

Our results were derived for simple static worlds. In future research we will explore the extent to which they also hold for more complex and dynamic worlds. Also, future research will aim to identify more conditions for tractability by exploring alternative ways to characterize an agent's sensory and perceptual complexity.

References

[1] R. C. Arkin, *Behavior-Based Robotics*. Cambridge, MA: The MIT Press, 1998.

[2] H. Hexmoor and D. Kortenkamp, "Issues on building software for hardware agents," *Knowledge Engineering Review*, vol. 10, no. 3, pp. 301–304, 1995.

[3] D. Lyons and A. Hendricks, "Planning as incremental adaptation of a reactive system," *Robotics and Autonomous Systems*, vol. 14, no. 4, pp. 255–288, 1995.

[4] J. Connell, "SSS: A hybrid architecture applied to robot navigation," in *Proceedings of the IEEE International Conference on Robotics and Automation*, Nice, France, 1992, pp. 2719–2724.

[5] B. Selman, "Near-optimal plans, tractability, and reactivity," in *Proceedings of the Fourth International Conference on Knowledge Representation and Reasoning (KR'94)*, Bonn, Germany, 1994, pp. 521–529.

[6] P. Jonsson, P. Haslum, and C. Bäckström, "Towards efficient universal planning: A randomized approach," *Artificial Intelligence*, vol. 117, pp. 1–29, 2000.

[7] P. E. Dunne, M. Laurence, and M. Wooldridge, "Complexity results for agent design," *Annals of Mathematics, Computing & Teleinformatics*, vol. 1, no. 1, pp. 19–36, 2003.

[8] M. Wooldridge, "The computational complexity of agent design problems," in *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS-2000)*, Boston, MA, 2000, pp. 341–348.

[9] I. A. Stewart, "The complexity of achievement and maintenance problems in agent-based systems," *Artificial Intelligence*, vol. 146, pp. 175–191, 2003.

[10] M. Wooldridge and P. E. Dunne, "The complexity of agent design problems; determinism and history dependence," *Annals of Mathematics and Artificial Intelligence*, vol. 45, pp. 343–371, 2005.

[11] R. A. Brooks, "A robust layered control system for a mobile robot," *IEEE Journal of Robotics and Automation*, vol. RA-2, no. 1, pp. 14–23, 1986.

[12] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA: W.H. Freeman, 1979.

[13] R. Downey and M. Fellows, *Parameterized Complexity*. Berlin: Springer, 1999.

[14] L. Fortnow, "The Status of the P Versus NP Problem," *Communications of the ACM*, vol. 52, no. 9, pp. 78–86, 2009.

[15] R. Niedermeier, *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.

[16] C. Sloper and J. A. Telle, "An overview of techniques for designing parameterized algorithms," *Computer Journal*, vol. 51, no. 1, pp. 122–136, 2008.

[17] M. Cesati. (2006) Compendium of Parameterized Problems. [Online]. Available: http://bravo.ce.uniroma2.it/home/cesati/research/compendium/

[18] A. Pnueli and P. Rosner, "On the synthesis of a reactive module," in *Proceedings of the Sixteenth ACM Symposium on the Principles of Programming Languages (POPL)*, 1989, pp. 179–190.

[19] R. A. Brooks, "A robot that walks: Emergent behavior from a carefully evolved network," *Neural Computation*, vol. 1, no. 2, pp. 253–262, 1989.

[20] ——, "Intelligence without representation," *Artificial Intelligence Review*, vol. 47, pp. 139–160, 1991.

[21] W. Haselager, J. van Dijk, and I. van Rooij, "A lazy brain? Embodied embedded cognition and cognitive neuroscience," in *Handbook of Cognitive Science: An Embodied Approach*, P. Calvo and T. Gomila, Eds. Elsevier, 2008, pp. 273–290.

[22] K. Koffka, *Principles of Gestalt Psychology*. New York: Harcourt Brace, 1935.

[23] P. van der Helm, "Dynamics of Gestalt psychology (Invited review of *perceptual dynamics: theoretical foundations and philosophical implications of gestalt psychology* by F. Sundqvist)," *Philosophical Psychology*, vol. 19, no. 1, pp. 274–279, 2006.