Research Article *Open Access*

Journal of Pioneering Artificial Intelligence Research

ISSN: 3069-0846

doi.org/10.63721/26JPAIR0153

SPARK and the Dynamic Behavioral Architecture of Autonomous Robots

Huseyin Murat Cekirge

Department of Mechanical Engineering, The City College of New York (CUNY), New York, USA

Citation: *Huseyin Murat Cekirge (2026) SPARK and the Dynamic Behavioral Architecture of Autonomous Robots. J of Poin Artf Research 2(5), 1-23. WMJ-JPAIR-153*

***Corresponding author**: Huseyin Murat Cekirge, Department of Mechanical Engineering, The City College of New York (CUNY), New York, USA. E-mail: hmcekirge@usa.net

Submitted: 24.08.2026 **Accepted**: 28.08.2026 **Published**: 07.09.2026

Keywords: Autonomous Robotics, Dynamic Behavioral Architecture, SPARK, Behavioral Functions, Internal and External Variables, Continuous Updating, Dynamic Matching, Behavioral Landscape, Action List, Action Selection, Robot Learning, Goal Emergence, Behavioral Adaptation

Introduction

Autonomous robotics has traditionally been concerned with the ability of a machine to perceive its environment, make decisions, and act without continuous human intervention. Classical autonomous systems may react to changing circumstances, select among alternative actions, navigate uncertain environments, and modify their behavior according to sensory information. Behavior-based robotics further demonstrated that useful intelligent behavior may arise through continuous interaction between perception and action rather than through a single centralized representation of the world [1]. Modern robotics has subsequently developed increasingly sophisticated architectures for perception, planning, learning, control, and human–robot interaction [2].

Nevertheless, autonomy does not necessarily imply that the purpose of the robot itself is autonomously generated. In many autonomous systems, a mission, task, objective, reward structure, or operational role exists before the autonomous decision process begins. The robot may determine how, when, and under what circumstances to act, while the general reason for acting remains externally established. Developmental robotics and intrinsic-motivation research have substantially expanded this picture by investigating systems capable of exploration, curiosity-driven learning, skill acquisition, and self-generated goals [3-5]. Autotelic-agent research goes further by considering agents capable of representing, generating, selecting, and solving their own problems rather than operating exclusively within externally specified individual tasks [6].

These developments raise a more fundamental behavioral question: **what causes a possible behavior or goal to emerge in the first place?** Existing work on developmental value systems has shown that accumulated experience, intrinsic motivation, attention,

acquired values, and innate or initialized values can influence future robotic behavior [7]. However, the present paper approaches the problem from a different engineering direction. Rather than beginning with a prescribed objective or immediately defining a reward or optimization function, the objective is to describe the continuously changing conditions under which an internal possibility may become behaviorally significant.

Consider two robots. The first is an autonomous combat robot assigned a mission. Its environment may change continuously, and the robot may independently determine its immediate actions. Nevertheless, the general mission precedes those decisions:

IMPOSED PURPOSE → CHANGING CONDITIONS → AUTONOMOUS DECISION → ACTION

Now consider a robot that encounters a tennis court and develops the initiative to play tennis, although no human has issued the command *play tennis*. If the possibility of playing tennis arises through the interaction of the robot's accumulated experience, internal state, external circumstances, location, history, and available behavioral knowledge, the direction of causality is different:

CONTINUOUSLY UPDATED STATE → EMERGING POSSIBILITY → CANDIDATE GOAL → AUTONOMOUS ACTION

The distinction is therefore not simply between a non-autonomous and an autonomous robot. Both robots may possess substantial autonomy. The distinction concerns the **origin of behavioral direction**. In the first case, autonomy operates under an imposed purpose. In the second, an initiative or candidate purpose may itself emerge from the evolving behavioral state of the robot. This distinction is consistent with the central question developed in this paper: a target need not always be an initial condition; under some circumstances it may become an output of the behavioral process.

To investigate this problem, the paper introduces the concept of a **SPARK**. SPARK is not initially assumed to be a particular equation, threshold, optimization criterion, or stochastic event. It denotes the presently unresolved event through which continuously changing internal and external conditions become sufficiently significant to initiate a new behavioral possibility. The distinction between matching, SPARK, and decision is deliberately maintained: matching describes compatibility among conditions, SPARK initiates a potentially meaningful new process, and decision determines whether the resulting candidate state or action is ultimately accepted.

The proposed architecture is dynamic. Internal and external variables, their relative weights, and their matching relationships may depend upon both the state and history of the agent and the time and location of an event. Matching itself need not be restricted to instantaneous values; compatibility of rates of change and higher-order changes may also carry behavioral information. Accordingly, function, first-derivative, and second-derivative matching are treated as potentially independent dimensions rather than mandatory simultaneous conditions. The resulting behavioral state is therefore continuously evolving rather than represented by a fixed behavioral map.

A practical architecture must also recognize that no robot requires every possible behavioral function. Robot classification determines the capacity of its behavioral library and consequently the internal and external variables that are available for behavioral evaluation. A single robot interacting with a human, for example, need not contain the same interaction functions as a cooperative multi-robot system. The architecture is therefore formulated as open: behavioral libraries may be extended, reduced, activated, or deactivated as the operational classification of the robot changes, without requiring reconstruction of the fundamental behavioral mechanism.

The engineering problem considered in this paper can consequently be summarized by the following progression:

ROBOT CLASSIFICATION → LIBRARY CAPACITY → INTERNAL/EXTERNAL VARIABLES → CONTINUOUS UPDATING → DYNAMIC MATCHING → BEHAVIORAL LANDSCAPE → SPARK → CANDIDATE GOAL / INITIATIVE → ACTION

The purpose of the paper is not to claim that the complete mechanism of autonomous goal formation has been solved. Nor is it to prescribe detailed behavioral functions for every possible robot classification. Instead, the objective is to establish an open dynamic behavioral architecture in which the origin, evolution, and possible emergence of behavioral initiative can be studied as an engineering problem.

Dimensionality of the Behavioral Domain

The behavioral state of an autonomous robot cannot, in general, be described solely by the present physical time and location of an observed event. The robot itself has a spatial position and an accumulated history, while an event has its own location and occurrence time. These quantities should therefore be distinguished explicitly.

The behavioral domain is represented as

$$\Omega = (X_r, X_e, T_r, T_e)$$

where

X_r = robot spatial coordinate or location

X_e = event spatial coordinate or location

T_r = robot time, representing age, accumulated operational history, or experience

T_e = event time measured on the external or absolute time scale

Thus, two different notions of time are present. T_e describes when an event occurs, whereas T_r describes where the robot stands in its own accumulated history when that event occurs. Similarly, X_r and X_e distinguish the location of the robot from the location associated with the event.

This distinction is important because the same external event need not have the same behavioral significance for the same robot at different stages of its operational history. Likewise, an identical event occurring at another location may produce a different behavioral response. The behavioral domain is therefore treated here as a four-dimensional domain consisting of two spatial and two temporal coordinates:

$$\Omega = (X_r, X_e, T_r, T_e)$$

The internal behavioral functions of the robot may consequently be expressed generally as

$$I_i = I_i(X_r, X_e, T_r, T_e), \quad i = 1, 2, \ldots, n$$

while the external behavioral functions may be represented as

$$E_j = E_j(X_r, X_e, T_r, T_e), \quad j = 1, 2, \ldots, m.$$

The internal functions I_i may represent, for example, operational need, energy state, safety state, accumulated experience, preference, mission influence, or other behavioral quantities available within the robot's library. Their detailed forms are not prescribed here. The set of available internal functions depends upon the robot classification and its library capacity.

Similarly, E_j represents externally originating behavioral information available to the robot. Depending upon its classification, this may include environmental conditions, detected events, objects, human interaction, commands, warnings, or other admissible external information. A robot cannot behaviorally evaluate an interaction domain that lies outside the capacity of its library.

Accordingly, the complete behavioral state may be written provisionally as

$$Z(\Omega) = [I_1(\Omega), I_2(\Omega), \ldots, I_n(\Omega), E_1(\Omega), E_2(\Omega), \ldots, E_m(\Omega)],$$

with additional weights, matching quantities, and SPARK-related conditions introduced subsequently.

The essential consequence is that the behavioral functions are not fixed constants. As the robot moves, accumulates experience, and encounters events at different places and times, one or more components of the behavioral domain may change:

$$\Delta\Omega = (\Delta X_r, \Delta X_e, \Delta T_r, \Delta T_e)$$

Consequently, in general,

$$\Delta I_i \neq 0 \text{ and/or } \Delta E_j \neq 0$$

The behavioral state is therefore continuously updated over the robot's operating history. The same robot confronted by an apparently identical event may occupy a different behavioral state because its location, accumulated history, or the space-time circumstances of the event have changed. This four-dimensional dependence is the foundation for the dynamic matching

and SPARK formulation developed in the following sections.

Throughout this paper, the following notation is retained:
X_r = robot location
X_e = event location
T_r = robot time (age, accumulated history, or experience)
T_e = absolute/event time

Internal and External Behavioral Functions
Within the behavioral domain defined by
$\Omega = (X_r, X_e, T_r, T_e)$

The state of the robot is represented by two principal groups of behavioral functions: internal functions and external functions. These functions are not assumed to be constant. Their values may change with robot location, event location, robot history, and event time. This space–time dependence is fundamental to the dynamic behavioral architecture.

The internal behavioral functions are written as
$i = 1, 2, \ldots, n \quad I_i(\Omega) = I_i(X_r, X_e, T_r, T_e)$

They describe behavioral quantities belonging to the robot itself. Depending upon the robot and its operational classification, examples may include
$I_1(\Omega)$ = energy or operational need
$I_2(\Omega)$ = safety or self-protection state
$I_3(\Omega)$ = accumulated experience
$I_4(\Omega)$ = learned preference or affinity
$I_5(\Omega)$ = mission or command influence
$I_6(\Omega)$ = curiosity, interest, or exploratory tendency

These examples are illustrative rather than prescriptive. The architecture does not require that every robot possess every internal function. In particular, the detailed mathematical forms of the individual functions are not specified for every robot classification.

A useful distinction is between a universal internal core and classification-dependent internal functions. Certain operational needs may exist regardless of whether the robot operates alone, with a human, or as part of a group. Other functions exist only because a particular interaction capability has been assigned to the robot. Thus, self-related operational functions may remain part of the universal core, whereas human–robot or robot–robot interaction functions may belong to classification-dependent libraries. This distinction allows the behavioral architecture to remain general without requiring every possible behavioral function to be installed in every robot.

External behavioral functions are represented by
$j = 1, 2, \ldots, m \quad E_j(\Omega) = E_j(X_r, X_e, T_r, T_e)$

These functions describe information originating outside the robot. Examples may include environmental conditions, detected objects, human actions, commands, warnings, opportunities, obstacles, or other events that are meaningful within the robot's operational domain.

It is important to distinguish an external function from raw sensory data. A camera image, microphone signal, temperature measurement, or position measurement is not necessarily itself a behavioral function. Sensory information becomes behaviorally relevant when it is interpreted within a domain represented in the robot's library.

Consequently,
SENSORY INPUT → INTERPRETATION → $E_j(\Omega)$
rather than assuming that every incoming sensor value directly participates in behavioral determination.

The internal and external functions can therefore be collected into the behavioral state vector
$Z(\Omega) = [I_1(\Omega), \ldots, I_n(\Omega), E_1(\Omega), \ldots, E_m(\Omega)]$.

The dimension of Z is not universal. It depends upon the behavioral capacity assigned to the particular robot. This leads directly to the concept of library capacity. The classification of a robot determines which behavioral domains it is capable of representing and therefore which internal and external functions need to exist:

ROBOT CLASSIFICATION → LIBRARY CAPACITY → AVAILABLE BEHAVIORAL FUNCTIONS

For example, consider a single autonomous robot designed to interact with one human but not behaviorally with other robots. Its library may contain,

L^c = {Self, Environment, Human}

Such a robot may require internal and external functions associated with human commands, requests, warnings, or observed human conditions. It does not, however, require behavioral functions for robot–robot negotiation, cooperative task allocation, herd behavior, or collective decision making. Another robot may physically appear in its sensory field and be treated as an environmental object without activating a robot–robot behavioral interaction mechanism.

Thus, a fundamental reduction is obtained:
$\mathbf{C \rightarrow L^c(C) \rightarrow \{I_i, E_j\}}$

The classification C determines library capacity $L^c(C)$, and the library capacity determines the set of behavioral functions available to the robot. A robot therefore need not evaluate behavioral domains that do not belong to its operational configuration. Furthermore, the complete library need not be active continuously. If $L^c(C)$ denotes the total behavioral library capacity and $L^A(T_e)$ denotes the portion currently active, then

$$\mathbf{L^A(T_e) \subseteq L^c(C)}$$

Library capacity describes what the robot can behaviorally represent, whereas the active library describes what is behaviorally relevant at the present state.

This distinction substantially reduces the size of the continuously evaluated behavioral problem. Rather than updating every possible behavioral function, only the functions associated with the active library need participate in the current state.

Accordingly, the active behavioral state may be represented as
$\mathbf{Z^A(\Omega) = [I_1^A(\Omega), \ldots, I_p^A(\Omega), E_1^A(\Omega), \ldots, E\_q^A(\Omega)]}$

where $p \leq n$ and $q \leq m$

As circumstances change, functions may enter or leave the active state. The behavioral architecture therefore requires a continuous updating procedure through which the relevant internal and external functions, and subsequently their weights and matching conditions, evolve with the robot and its environment.

This produces the transition to the next stage of the architecture:

Internal Functions + External Functions
→ Active Behavioral State
→ Continuous Updating
→ Dynamic Matching
→ SPARK

The internal and external functions therefore do not directly prescribe behavior. They establish the continuously changing behavioral state from which matching, SPARK, initiative, and eventual action may emerge.

Robot Classification and Library Capacity

The proposed behavioral architecture is intended to remain open and applicable to different classes of autonomous robots. However, it is neither necessary nor computationally desirable for every robot to contain behavioral functions associated with every possible form of interaction. A robot designed to operate alone, for example, does not necessarily require the behavioral functions needed for robot–robot cooperation. Similarly, a robot that does not interact with humans does not require the same human-interaction behavioral library as a robot operating continuously with a human partner.

A preliminary classification can therefore be established according to three characteristics: **organization**, **human interaction**, and **authority relationship**.

Table 1: Robot Classification and Corresponding Behavioral Library Capacity.

Class	Organization	Human Contact	Authority	Behavioral Library
C_1	Single	No	—	Self + Environment
C_2	Single	Yes	Human-Master	Self + Environment + Human
C_3	Single	Yes	Peer/Cooperative	Self + Environment + Human-Cooperation
C_4	Single	Yes	Robot-Master	Self + Environment + Human-Management
C_5	Herd	No	—	Self + Environment + Herd
C_6	Herd	Yes	Human-Master	Self + Environment + Herd + Human
C_7	Herd	Yes	Peer/Cooperative	Self + Environment + Herd + Human-Co-operation
C_8	Herd	Yes	Robot-Master	Self + Environment + Herd + Human-Man-agement

This classification is not intended to constitute an exhaustive taxonomy of autonomous robots. Other organizational structures, authority relationships, and interaction domains are possible. The purpose of the classification is instead to establish the **behavioral capacity required by a particular robot** while maintaining an open architecture.

The relationship can be expressed conceptually as
Robot Classification → Library Capacity → Available Behavioral Functions

If $\mathbf{L^c(C)}$ denotes the behavioral library capacity associated with robot class **C**, the present study may, for example, consider the class

C_2 = Single Robot + Human Interaction + Human-Master Authority

Its behavioral library may then be represented as
$\mathbf{L^c(C_2) = \{L^I, L^E, L^H\}}$

where $\mathbf{L^I}$ represents the self/internal domain, $\mathbf{L^E}$ the environmental domain, and $\mathbf{L^H}$ the human-interaction domain.

A robot belonging to this class does not require behavioral functions associated with herd behavior or robot–robot cooperation. Another robot may physically enter its sensory field and still be treated simply as part of the external environment if robot–robot behavioral interaction is not included in its library. Thus, classification determines not only what information the robot may receive, but also **which behavioral functions need to exist within the architecture**. This principle is consistent with the previously developed relationship between classification, library capacity, and available behavioral variables.

The set of available internal and external functions is therefore classification dependent. The functions themselves retain the four-dimensional dependence introduced previously,

$\mathbf{\Omega = (X_r, X_e, T_r, T_e)}$

while the robot classification **C** determines which of these functions are available within the behavioral library. Conceptually, this relationship may be represented as $\mathbf{\{I_i(\Omega), E_j(\Omega)\} = F(C)}$

The complete library capacity need not be active at every instant. If $\mathbf{L^A(T_e)}$ denotes the currently active library,
$\mathbf{L^A(T_e) \subseteq L^c(C)}$
Only the behavioral functions associated with the active domains need to enter the current updating procedure.

Library capacity therefore describes what the robot **can behaviorally represent**, while the active library describes what the robot is **currently required to evaluate**. This distinction substantially reduces the continuously evaluated behavioral problem.

The resulting architecture becomes
CLASSIFICATION → LIBRARY CAPACITY → ACTIVE LIBRARY → ACTIVE INTERNAL/ EXTERNAL FUNCTIONS → CONTINUOUS UPDATING → MATCHING → BEHAVIORAL LANDSCAPE → SPARK

This provides a short computational route through what would otherwise become an extremely large behavioral program. The robot is not required to evaluate interaction domains that do not belong to its operational configuration.

At the same time, the classification is not necessarily permanent. Because the architecture is open, a robot may subsequently be **upgraded, downgraded, promoted, or demoted** by changing its authorized library capacity. A single human-interacting robot, for example, may later acquire a robot–robot cooperative library without requiring the fundamental behavioral architecture to be redesigned.

For the remainder of the present formulation, however, only one case needs to be examined in detail:
Single Robot + Human Interaction + Human-Master Authority
→ Library Capacity
→ Active Behavioral Functions
→ Continuous Updating
→ Dynamic Behavioral Landscape
→ SPARK

The remaining classes demonstrate the openness and extensibility of the architecture but do not require detailed behavioral-function formulations in the present paper.

Establishing the Behavioral Functions

The behavioral functions of a robot may originate from different sources. Some may be established during design and initialization, some may be acquired through training and accumulated experience, and others may develop from the continuing interaction of the robot with its environment. Human interaction may provide an additional source when such interaction is included within the classification and library capacity of the robot. The purpose of the present formulation is not to prescribe how every individual behavioral function must be created. Rather, it is assumed that the functions required by the robot's classification are available within its behavioral architecture and can subsequently be updated as conditions change.

A master's command provides a useful example. The command initially originates outside the robot and enters through the human–robot interaction channel. Once the command has been received, interpreted, and retained, however, its behavioral representation exists within the robot. From that point onward, the command is treated as **one of the internal behavioral functions**. It is not a separate decision mechanism, nor does its presence by itself determine the complete behavior of the robot. This treatment is consistent with the proposed architecture, in which command, accumulated experience, learned preference, internal needs, and other internal quantities may simultaneously contribute to the behavioral state.

For example, a robot may receive the command *remain here*. The command remains represented internally while the robot continues to observe its environment and update its other behavioral functions. If circumstances remain unchanged, the command may continue to support the existing behavior. If a significant new event occurs, however, such as the detection of a person in immediate danger, the environmental and internal states of the robot change. The original command has not disappeared; it remains one of the active internal functions. Nevertheless, it now participates together with safety-related functions, accumulated experience, environmental information, and other relevant behavioral quantities. The previously developed example illustrates precisely this coexistence between a retained command and an emerging initiative.

This distinction is particularly important for an autonomous behavioral architecture. A robot capable of accepting commands need not be reduced to a command-executing machine. Conversely, autonomy does not require that commands be excluded. A command may enter the system, become internally represented, and participate in the same continuously changing behavioral structure as the robot's other

functions. Whether the resulting behavior follows the command, modifies an existing course of action, or produces a new initiative depends upon the subsequent interaction among the active behavioral functions and the conditions developed later in the SPARK mechanism.

The detailed mathematical formulation of human–robot interaction is beyond the scope of this paper. The master's command is introduced only as an example demonstrating how an externally originating event may become an internal behavioral function. Other functions may be established through different mechanisms. What is essential for the present architecture is that, once established, these functions become components of the robot's behavioral state and are subsequently subject to the continuous updating process.

SPARK and Levels of SPARK

The continuously updated internal and external behavioral functions provide the conditions from which a **SPARK** may emerge. A SPARK is not defined here as the final decision or action. It represents the behavioral event at which one or more relationships among the active functions become sufficiently significant to initiate a new behavioral possibility. Matching, SPARK, and decision are therefore distinct stages. Matching describes the relationship among functions; SPARK initiates a potentially meaningful behavioral process; and a subsequent decision determines whether the emerging possibility becomes an accepted goal or action.

Because the behavioral functions are continuously changing, matching need not occur only between their instantaneous values. Their rates of change and changes in those rates may also contain behavioral information. Accordingly, three basic matching components are considered:

$$M_0 = \text{Match}(F_1, F_2)$$
$$M_1 = \text{Match}(F_1', F_2')$$
$$M_2 = \text{Match}(F_1'', F_2'')$$

where M_0 represents function or state matching, M_1 represents first-derivative or rate-of-change matching, and M_2 represents second-derivative or change-of-rate matching. These components form a multidimensional matching state.

$$M = [M_0, M_1, M_2]$$

The significance of derivative matching is that two behavioral functions may temporarily possess similar values while evolving in entirely different directions. Conversely, two functions that are not presently equal may be dynamically approaching one another. The first and second derivatives therefore provide information about the evolving relationship between behavioral functions that cannot necessarily be obtained from their instantaneous values alone.

Two behavioral states need not be strongly matched at a particular instant for a significant relationship to be developing. Their rates of change may indicate that they are approaching one another, while their second derivatives may indicate whether this convergence is strengthening or weakening. Conversely, two states that presently exhibit strong matching may already be dynamically separating. Behavioral significance may therefore reside not only in the present state, but also in the direction and acceleration of its evolution.

Level 0 SPARK — State Matching

The simplest SPARK may arise through sufficient matching of the behavioral functions themselves. In this case,

$$M_0 \rightarrow \text{SPARK}$$

This may represent a relatively immediate coincidence or compatibility between an internal condition and an external circumstance. No derivative matching is necessarily required.

Level 1 SPARK — Dynamic Matching

A second possibility involves the rates of change of the behavioral functions:

$$M_1 \rightarrow \text{SPARK}$$

or, depending upon the circumstances,

$$M_0 + M_1 \rightarrow \text{SPARK}$$

Here the behavioral significance is associated not only with the current states but also with how those states are

changing. The interaction therefore contains dynamic rather than purely instantaneous information.

Level 2 SPARK - Acceleration Matching

A further level may involve second-derivative compatibility:
$\mathbf{M_2 \rightarrow SPARK}$

or combinations such as
$\mathbf{M_1 + M_2 \rightarrow SPARK}$
and
$\mathbf{M_0 + M_1 + M_2 \rightarrow SPARK}$

Second-derivative matching introduces information concerning how the rates themselves are changing and may therefore identify behavioral relationships that are not visible from function values or first derivatives alone.

These SPARK levels do **not** constitute a mandatory hierarchy. A Level 2 SPARK does not necessarily require prior satisfaction of Levels 0 and 1. Depending upon the robot, the active behavioral functions, and the circumstances, one, two, or all three matching components may participate. Thus, the required number of matching conditions may itself be state dependent.

More generally, the active matching structure may be represented by

$\mathbf{S_a = S_a(\Omega, R)}$
and
$\mathbf{N_m = N_m(\Omega, R)}$

where $\mathbf{S_a}$ represents the active set of matching conditions, $\mathbf{N_m}$ represents the number of conditions required under the current circumstances, **R** identifies the robot, and $\mathbf{\Omega = (X_r, X_e, T_r, T_e)}$ represents the behavioral domain. The underlying material explicitly allows both the active matching conditions and their required number to depend upon the agent and its space–time circumstances.

Consequently, the same robot need not require the same SPARK level throughout its operational life. A particular relationship between an internal need and an external event may require only state matching at one time and location, while a similar event encountered at another stage of the robot's history may require additional dynamic matching. The SPARK conditions therefore belong to the changing behavioral landscape rather than to a permanently fixed decision table.

At this stage, the exact transformation from matching to SPARK is deliberately left open:
$\mathbf{[M_0, M_1, M_2, \ldots] \rightarrow ? \rightarrow SPARK}$

No universal threshold, optimization procedure, probability distribution, or deterministic equation is imposed prematurely. The present purpose is to establish the quantities and dynamic relationships from which such a mechanism can subsequently be investigated. This unresolved transition is explicitly preserved in the developing formulation.

Thus, a SPARK may be viewed as a **multidimensional, state-dependent behavioral activation event**. Its level and required matching conditions may change with the robot, its accumulated history, its location, the external event, and the continuously evolving internal and external behavioral functions.

Dynamic Behavioral Landscape and Dimensional Reduction

The behavioral architecture introduced in the preceding sections is defined over the four-dimensional domain.

$\mathbf{\Omega = (X_r, X_e, T_r, T_e)}$

where $\mathbf{X_r}$ and $\mathbf{T_r}$ describe the robot's location and accumulated robot time, while $\mathbf{X_e}$ and $\mathbf{T_e}$ describe the spatial and temporal coordinates associated with an event. Internal functions, external functions, their weights, and their matching relationships may therefore change throughout this domain. The resulting behavioral representation should not be interpreted as a fixed map. It is a dynamic landscape that changes as the robot moves, accumulates experience, and encounters new circumstances. This dependence of the internal and external variables and matching structure upon the four-dimensional operating domain is already inherent in the formulation.

Although the complete architecture is four-dimensional, the instantaneous behavioral problem may be considerably smaller. At a particular present state, the robot occupies a known location and a known point in

its operational history. These quantities may therefore be held fixed:

$\mathbf{X_r = X_r^*}$
$\mathbf{T_r = T_r^*}$

while the location and time associated with an actual, anticipated, or imagined event remain variable. A general matching representation $\mathbf{M = M(X_r, X_e, T_r, T_e)}$ can consequently be reduced at the present robot state to $\mathbf{M = M(X_e, T_e \mid X_r^*, T_r^*)}$.

Thus, the architecture remains four-dimensional, while the instantaneous event field may be examined as a conditional two-dimensional landscape.

This distinction has a useful behavioral interpretation. The robot evaluates possibilities from **where it is now and what it has become up to now**. Its present robot state is fixed for the current evaluation, while possible events may occupy different locations and times. As the robot subsequently moves and its operational history increases, $\mathbf{(X_r^0, T_r^0) \rightarrow (X_r^1, T_r^1) \rightarrow (X_r^2, T_r^2) \rightarrow \ldots}$ a new conditional event landscape is produced. Consequently, even if the external circumstances appear similar, the behavioral landscape need not remain the same.

The conditional landscape may contain different local geometric structures. Depending upon the behavioral functions and their interactions, regions resembling a **valley, peak, saddle, ridge, or other local configuration** may appear. Such terminology is used here to characterize the geometry of the behavioral landscape rather than to assert that the robot is solving an optimization problem.

This distinction is essential. The existence of a multidimensional behavioral surface does not imply that behavior must be generated by minimization or maximization. A SPARK might eventually be associated with a minimum, maximum, saddle region, transition region, or another presently unidentified feature of the landscape. At this stage, no such mechanism is imposed.

The four-dimensional formulation does not imply an optimization process. Optimization or minimization may emerge as one possible mechanism for behavioral selection, but this is not assumed a priori.

Accordingly, the present formulation retains the more general relationship
Matching → Unknown Behavioral Mechanism → SPARK

rather than prematurely imposing

Matching → Optimization → SPARK

This distinction is particularly important because the matching process itself may involve function values, first derivatives, second derivatives, or combinations of these quantities. The developing formulation explicitly allows these matching dimensions to operate independently rather than requiring simultaneous satisfaction of all derivative levels.

For illustration, a stationary point of a conditional matching surface might satisfy
$\mathbf{\partial M/\partial T_e = 0 \text{ and } \partial M/\partial X_e = 0}$

Such a point could be a minimum, maximum, or saddle point. For example, the condition
$\mathbf{D = (\partial^2 M/\partial X_e^2)(\partial^2 M/\partial T_e^2) - (\partial^2 M/\partial X_e \partial T_e)^2 < 0}$

would identify a saddle point of a sufficiently smooth two-dimensional surface. However, the present theory does **not** assert that SPARK occurs at such a stationary point. The example merely demonstrates that a behavioral landscape may possess meaningful local geometry without requiring that the behavioral mechanism be defined as optimization.

The important result is therefore the dimensional distinction:
GLOBAL BEHAVIORAL ARCHITECTURE: 4D

$\mathbf{\Omega = (X_r, X_e, T_r, T_e)}$

while, at a fixed present robot state,
INSTANTANEOUS EVENT LANDSCAPE: potentially 2D
$\mathbf{M(X_e, T_e \mid X_r^*, T_r^*)}$

As the robot moves through space and accumulates operational history, this conditional landscape is

continuously reconstructed. SPARK therefore emerges, if its conditions are satisfied, not within a permanent behavioral map but within a continuously deforming behavioral landscape.

Two Illustrative Examples

The distinction between autonomous execution and the autonomous emergence of a behavioral initiative can be illustrated by considering two robots operating under substantially different conditions. The purpose of these examples is not to prescribe detailed behavioral functions, but to demonstrate how the same dynamic behavioral architecture may accommodate both an externally imposed mission and an internally emerging candidate goal.

Combat Soldier Robot

Consider an autonomous combat soldier robot assigned a specific mission. The mission, operational constraints, and applicable commands are established externally. The robot may nevertheless possess considerable autonomy in determining how to execute the assigned mission under changing environmental conditions.

Its general behavioral sequence may be represented as **IMPOSED MISSION → CONTINUOUS UPDATING → MATCHING → BEHAVIORAL LANDSCAPE → SPARK → CANDIDATE LOCAL INITIATIVE**

The presence of an imposed mission does not imply that every action of the robot must be explicitly commanded. Environmental conditions may change rapidly, requiring the robot to interpret new events, update its internal and external behavioral functions, and initiate actions that were not individually specified in advance.

A command received from an authorized human enters from an external source. Once interpreted and retained, however, its behavioral representation becomes one of the robot's internal functions. It subsequently exists together with other internal quantities such as operational needs, accumulated experience, safety-related functions, and mission-related states. The command therefore influences behavior without constituting the entire behavioral mechanism. This treatment follows the earlier formulation in which command participates together with experience, learned preference, and other internal variables.

Suppose, for example, that the robot is instructed to maintain a particular position. While carrying out this command, an unexpected event occurs. The external behavioral functions change, the active internal functions are updated, and new matching conditions may develop. A SPARK may then generate a local initiative concerning how the existing mission should be executed. However, the fundamental purpose has not been generated by the robot.

The important characteristic of this case is therefore: **The Robot Autonomously Determines How to Act, While the General Purpose of the Action Remains Externally Imposed.**

This represents autonomy within an assigned purpose.

The Robot That Wants to Play Tennis

Now consider a substantially different situation.

An autonomous robot has not been instructed to play tennis. No tennis mission has been assigned, and play tennis does not initially exist as the current goal.

The robot nevertheless possesses an appropriate library capacity containing concepts and behavioral functions associated with physical activity, tennis, human interaction, previous experience, available energy, curiosity, preference, or other relevant quantities. The exact functions need not be prescribed for the example. At a particular robot state, (X_r^*, T_r^*) the robot encounters circumstances associated with tennis. It may observe a tennis court, people playing, available equipment, or another combination of events represented within its library.

These external events enter the continuously updated behavioral state. At the same time, relevant internal functions such as accumulated experience, present operational needs, preference, curiosity, or previous association with the activity—possess their own current values and rates of change.
The process may therefore develop as

Observation / Event
→ Continuous Updating
→ Activation Of Relevant Library Functions
→ Dynamic Matching
→ Behavioral Landscape
→ SPARK
→ "Play Tennis" As A Candidate Goal

Here the essential difference is that **play tennis was not supplied as the initial objective**. It appears downstream of the behavioral process.

The robot has therefore not merely answered the question
"How should I play tennis?"

It has first generated a more fundamental possibility:
"I could play tennis."

Whether that possibility ultimately becomes an accepted goal and produces action is a subsequent question. SPARK does not necessarily guarantee execution. It may generate a candidate initiative that is later accepted, rejected, postponed, or displaced by another behavioral requirement.

Thus,
SPARK → CANDIDATE GOAL ≠ AUTOMATIC ACTION

This distinction is consistent with the developing formulation in which SPARK initiates a potentially meaningful process but remains separate from the final decision.

Comparison of the Two Cases

The two robots may both legitimately be described as autonomous. The fundamental distinction is therefore not simply autonomous versus non-autonomous.

For the combat soldier robot:
Purpose imposed → autonomous initiative within purpose

For the tennis robot:
No corresponding present purpose → behavioral interaction → SPARK → candidate purpose emerges

The contrast may be summarized as:

Table 2: Comparison of Imposed-Purpose and Emerging-Purpose Autonomous Behavior.

Combat Soldier Robot	Tennis Robot
General mission externally imposed	Tennis goal not externally imposed
Goal exists before local behavioral processing	Candidate goal emerges from behavioral processing
SPARK may generate local initiative	SPARK may generate a new candidate goal
Autonomy primarily concerns execution	Autonomy extends toward goal emergence
"How should I accomplish the mission?"	"What might I choose to do?"

The second example therefore illustrates the central problem addressed by the SPARK architecture: **a goal need not always be an input to autonomous behavior; under appropriate conditions, a candidate goal**

may become an output of the behavioral process.

From SPARK to Action List

The occurrence of a SPARK does not necessarily determine what the robot will do. As introduced in the preceding sections, SPARK represents the emergence of a behaviorally significant possibility rather than the final decision or action. The next requirement is therefore to determine what actions are available to the robot under the conditions created by that SPARK. This leads to the formation of an **Action List**.

The Action List is the set of admissible actions that the robot is capable of considering at the current behavioral state. It may be represented as

$$A(S_p) = \{A_1, A_2, \ldots, A_k\}$$

where each A_k represents a candidate action available following the SPARK.

A SPARK may be associated with a previously established Stored Action Sublist containing actions relevant to that class of behavioral event. If S_p denotes the occurring SPARK, the corresponding stored action sublist may be represented as

$$A^s(S_p) = \{A_1, A_2, \ldots, A_k\}$$

This stored sublist does not constitute the final Action List. Its elements are subsequently evaluated against the robot's current classification, authority, capabilities, active behavioral state, environmental conditions, and applicable constraints. The admissible elements form the Active Action List.

$$A^A(S_p, \Omega) \subseteq A^s(S_p)$$

Thus, the robot need not construct or search the entire universe of possible actions following every SPARK. The SPARK identifies a relevant stored action domain, while the present behavioral state determines which elements of that domain are currently admissible.

The Action List is not universal and should not contain every action that the robot could theoretically perform. Its contents depend upon the robot classification, library capacity, present internal and external functions, current circumstances, and the behavioral direction activated by the SPARK. The same principle used earlier to reduce the behavioral library therefore applies to the action domain: only relevant and admissible actions need to enter the current list.

The process can be represented as

SPARK → STORED ACTION SUBLIST → STATE / AUTHORITY / CAPABILITY GATING → ACTIVE ACTION LIST

This distinction is important because a SPARK may be compatible with several different responses. For example, a SPARK associated with the detection of danger does not uniquely imply that the robot must immediately intervene. Depending upon its classification and authority, possible actions may include remaining in position, issuing a warning, requesting additional information, approaching the event, requesting human authorization, or taking immediate protective action. The SPARK establishes behavioral significance; it does not by itself prescribe which of these actions must be executed.

The combat soldier robot introduced earlier provides a useful example. Following a SPARK generated by changing mission and environmental conditions, an Action List might contain actions such as maintaining the assigned position, observing the developing event, warning the authorized human, changing position, or taking an immediate mission-related action. The mission and retained command remain part of the robot's behavioral state while the Action List represents the currently available alternatives. The preceding example already distinguishes the externally imposed general purpose from the robot's locally generated initiative.

The tennis robot provides a different case. A SPARK associated with tennis does not mean that the robot immediately begins playing. The resulting Action List may contain alternatives such as ignoring the possibility, continuing observation, approaching the tennis court, determining whether a partner is available, seeking authorization when required by its classification, or proceeding toward playing tennis. Thus, even when the SPARK contributes to the emergence of a new candidate goal, several possible actions may remain available.

An important distinction therefore follows:

SPARK ≠ ACTION
and

SPARK ≠ ACTION SELECTION

Instead,
SPARK → STORED ACTION SUBLIST → ACTIVE ACTION LIST → ACTION SELECTION → ACTION

The generation of the Action List and the selection of an action are consequently separate operations. The first determines **what the robot can presently do**; the second determines **which admissible action should be taken.**

The Action List may also contain a **no-action alternative**. A SPARK indicates that a behavioral possibility has become significant, but significance does not necessarily require immediate physical action. Waiting, continuing observation, retaining the emerging possibility, or requesting additional information may themselves constitute legitimate alternatives. This prevents SPARK from being artificially coupled to compulsory execution.

The detailed mechanism by which the final action is selected from the Action List is not imposed at this stage. Selection may depend upon behavioral priorities, constraints, safety requirements, authority relationships, computational considerations, or another decision mechanism developed for the robot. In particular, the existence of an Action List does not imply that optimization must be used.

The resulting behavioral sequence can therefore be extended to
CONTINUOUS UPDATING → MATCHING → DYNAMIC BEHAVIORAL LANDSCAPE → SPARK → ACTION LIST → ACTION SELECTION → ACTION

This additional stage preserves the central role of SPARK while clearly separating **the emergence of behavioral significance from the selection and execution of behavior**.

Position of the Action List in the Dynamic Behavioral Architecture

The Action List occupies a specific position within the proposed dynamic behavioral architecture. It is generated **after the occurrence of a SPARK and before the final action-selection process**. This position is important because SPARK, Action List generation, action selection, and action execution represent different behavioral operations and should not be combined into a single decision step.

The complete behavioral sequence may be represented as
CLASSIFICATION → LIBRARY CAPACITY → ACTIVE LIBRARY → DATA STREAM → INTERNAL AND EXTERNAL FUNCTIONS → CONTINUOUS UPDATING → DYNAMIC MATCHING → BEHAVIORAL LANDSCAPE → SPARK → ACTION LIST → ACTION SELECTION → ACTION → FEEDBACK

The first part of this sequence establishes the continuously changing behavioral state of the robot. Classification determines the available behavioral library, the active library determines the functions relevant to the present circumstances, and the continuous updating procedure maintains the current internal and external behavioral state. Matching among these functions contributes to the dynamic behavioral landscape from which a SPARK may emerge.

Once a SPARK occurs, the architecture enters a different stage. The question is no longer only whether a behaviorally significant relationship has emerged. The robot must now determine **what it can do in response to that SPARK**.

For this purpose, an **Action List Generator** is introduced between SPARK and action selection.

Its conceptual input consists of the current SPARK, the active behavioral state, the robot classification, and the available library capacity. Its output is a finite set of admissible candidate actions,

$$\mathbf{A = \{A_1, A_2, \ldots, A_k\}}$$

The Action List Generator does not necessarily invent completely new physical capabilities. Rather, it identifies the actions available to the robot that are relevant to the behavioral direction activated by the SPARK and permitted by the robot's classification,

authority, operational limitations, and available library.

This produces three distinct questions within the behavioral architecture.

SPARK: Has a behaviorally significant possibility emerged?
ACTION LIST: What admissible actions are available in response to it?
ACTION SELECTION: Which of these actions, if any, should be executed?

These questions should remain separate. A SPARK does not uniquely determine an action, and the existence of an admissible action does not imply that it must be selected.

For example, if a human-related danger produces a SPARK in a robot whose classification permits human interaction, the resulting Action List may include observation, warning, approaching the event, requesting authorization, intervention, or no immediate action. The precise contents of the list depend upon the capabilities and authority associated with that robot. A different robot classification may produce a different Action List from an otherwise similar SPARK.

The Action List therefore provides an important connection between the **behavioral architecture** and the robot's **physical and operational capabilities**. SPARK may establish a behavioral direction, but the Action List restricts the subsequent decision to actions that the robot can actually and permissibly perform.

After an action has been selected and executed, its consequences return to the architecture as new information. The action may change the robot's position, external environment, internal state, accumulated experience, or interaction with humans and other agents. The resulting information re-enters the continuous updating procedure.

The architecture consequently forms a closed behavioral loop:
SPARK → STORED ACTION SUBLIST → GATING → ACTIVE ACTION LIST → ACTION SELECTION → ACTION → NEW INTERNAL/ EXTERNAL STATE → CONTINUOUS UPDATING → MATCHING → BEHAVIORAL LANDSCAPE → NEW SPARK

The Action List is therefore not an appendix to the SPARK mechanism. It is a required intermediate layer separating the **emergence of behavioral significance** from the **selection and execution of physical behavior**.

Following a SPARK, the architecture retrieves the **Stored Action Sublist** associated with that SPARK or SPARK class. This stored sublist represents previously available action possibilities rather than the final set of actions admissible under the present circumstances. The retrieved actions are therefore passed through a gating process based on the robot's current behavioral state, classification, authority, physical capability, environmental conditions, and applicable constraints.

Stored Action Sublists Associated with SPARK

The occurrence of a SPARK does not require the robot to search its complete action capability in order to determine what may be done next. Such a procedure would require the repeated consideration of many actions that may have no relationship to the behavioral condition represented by the SPARK. Instead, the proposed architecture associates identifiable SPARK classes with corresponding **stored action sublists.**

Let the behavioral library contain a set of identifiable SPARK classes,
$\mathbf{S = \{S_1, S_2, \ldots, S_p\}}$

For each SPARK class $\mathbf{S_i}$, a corresponding set of potentially applicable actions may be stored as
$\mathbf{A(S_i) = \{A_{i1}, A_{i2}, \ldots, A_{ik}\}}$

The association $\mathbf{S_i \rightarrow A(S_i)}$ does not prescribe the action that the robot must perform. It establishes only the set of actions that may reasonably be considered when that particular class of SPARK occurs.

For example, a SPARK associated with a detected danger to a human may be linked to a stored action sublist containing such possibilities as continued observation, warning the human, requesting additional information, requesting assistance, approaching the event, intervention, or no immediate action. A SPARK associated with an emerging interest in playing tennis would activate a different sublist, potentially containing continued observation, approaching the court, identifying

an available partner, requesting participation when necessary, playing tennis, postponing the activity, or abandoning the possibility.

The stored sublist therefore acts as an intermediate structure between SPARK and the active Action List. When SPARK S_i occurs, its associated action sublist is retrieved:

SPARK S_i → RETRIEVE $A(S_i)$

However, the retrieved sublist is not necessarily identical to the final Action List. Some stored actions may be impossible, inappropriate, unauthorized, or irrelevant under the present circumstances. The retrieved sublist must therefore be conditioned by the current behavioral state, robot classification, authority, active library, physical capability, and applicable operational constraints.

The procedure becomes
SPARK
→ Stored Action Sublist
→ Current-State Gating
→ Active Action List
→ Action Selection
→ Action

If **$G[Z(\Omega), C]$** represents the current gating conditions determined by the behavioral state **$Z(\Omega)$** and robot classification **C**, the active Action List may be expressed conceptually as

$A^A(S_i, \Omega) = G[A(S_i), Z(\Omega), C]$

Thus, the stored action sublist describes **what may potentially be done in response to a particular SPARK**, whereas the active Action List describes **what may presently be done under the existing conditions**.

This distinction also connects the Action List directly to the previously introduced concept of library capacity. Robot classification determines the behavioral domains and capabilities represented within the robot. Consequently, different robot classes need not possess identical action sublists for the same general SPARK. A human-authority robot, a cooperative robot, and a robot operating without human interaction may have different admissible responses even when they encounter a similar external event.

The architecture therefore avoids two extremes. It does not impose a rigid relationship of the form **SPARK → PREDETERMINED ACTION** and it does not require an unrestricted search of the robot's complete action universe. Instead, it introduces a bounded intermediate structure:

SPARK → ASSOCIATED POSSIBILITIES → PRESENTLY ADMISSIBLE POSSIBILITIES → SELECTION

The stored SPARK–action associations may themselves be established during design, training, accumulated experience, or authorized updating. Their detailed learning and modification mechanisms are not prescribed in the present paper. This maintains the open character of the architecture while identifying where such information must reside.

Accordingly, the behavioral library contains not only information required for interpreting internal and external conditions, but also structured associations between recognizable SPARK classes and their possible behavioral responses. The SPARK therefore acts as an **index into a relevant region of the action library**, rather than as a direct command for action.

Action Selection

Once a SPARK has occurred and the corresponding stored action sublist has been retrieved and conditioned by the current behavioral state, the robot is presented with an **Active Action List**. The next problem is to determine which action, if any, should be executed.

Let the Active Action List be represented as
$A^A(S_i, \Omega) = \{A_1, A_2, \ldots, A_k\}$

The elements of **A^A** have already passed the preceding gating process. They are therefore actions that are presently available and admissible according to the robot classification, library capacity, authority, physical capability, and current behavioral conditions. Action selection operates only on this reduced set.

This distinction is important. The selection mechanism does not search the entire behavioral or physical

capability of the robot. That reduction has already occurred through the sequence
SPARK → Stored Action Sublist → Current-State Gating → Active Action List.

Action selection addresses a narrower question:
Given the presently admissible actions, which action should be taken?

The selection process may depend upon the same continuously updated internal and external behavioral functions that contributed to the formation of the SPARK. However, their importance at the action-selection stage need not be identical to their importance during SPARK formation. A function that contributed strongly to the emergence of a behavioral possibility may become less important when determining how that possibility should be acted upon. Conversely, safety, authority, energy, feasibility, time, or other constraints may become dominant during action selection.

A master's command provides a useful example. As established earlier, a retained command is one of the internal behavioral functions of the robot rather than a separate decision mechanism. It may therefore influence the relative suitability of actions in the Active Action List without automatically determining the selected action. In a command-responsive but initiative-capable robot, the selected action results from the current behavioral state rather than from a simple fixed relationship of the form **Command → Action**.

The selection mechanism can be represented generally as
$\mathbf{A^* = D(A^A, Z^A, C, \Omega)}$

where $\mathbf{A^*}$ is the selected action, $\mathbf{A^A}$ is the Active Action List, $\mathbf{Z^A}$ is the current active behavioral state, **C** is the robot classification, and **Ω** represents the current behavioral domain.

The function **D** is deliberately left general. At this stage, no particular optimization, minimization, probabilistic selection, voting mechanism, priority rule, or deterministic algebraic procedure is imposed. The existence of several candidate actions does not by itself imply that an optimization problem must be solved. This is consistent with the earlier treatment of the behavioral landscape, where multidimensional structure was not assumed a priori to require minimization or maximization.

Accordingly,
ACTION LIST → SELECTION MECHANISM → SELECTED ACTION
is preferred to the more restrictive assumption
ACTION LIST → OPTIMIZATION → SELECTED ACTION

Optimization may eventually prove appropriate for some robot classifications or behavioral situations. A priority rule, elimination procedure, deterministic comparison, learned selection mechanism, or another method may be preferable in others. The present architecture separates the **existence of the selection problem** from the **algorithm used to solve it**.

The possibility of selecting **no immediate physical action** should also be retained. Waiting, continuing observation, requesting additional information, postponing a candidate behavior, or maintaining the existing state may constitute valid outcomes. Thus, the Active Action List may explicitly contain $\mathbf{A_0}$ **= No Immediate Action.**

This prevents the occurrence of a SPARK from being interpreted as compulsory physical execution.

Action selection must also respect constraints that may override otherwise attractive alternatives. Physical impossibility, insufficient energy, safety restrictions, authority limitations, operational rules, or unavailable resources may remove an action before final selection. Such constraints should preferably operate during gating so that inadmissible alternatives do not remain in the Active Action List.

The complete post-SPARK process can therefore be represented as
SPARK
→ Stored Action Sublist
→ Current-State Gating
→ Active Action List
→ Action Selection
→ Selected Action
→ Execution

After execution, the selected action changes either the robot, its environment, or both. These changes generate new internal and external information and return to the Continuous Updating Procedure. The action-selection process is therefore embedded within the closed behavioral loop rather than representing the termination of the architecture:

ACTION → NEW DATA → CONTINUOUS UPDATING → MATCHING → BEHAVIORAL LANDSCAPE → SPARK → NEW ACTION LIST → NEW SELECTION

This separation of stages is fundamental. **SPARK determines that a behavioral possibility has become significant; the stored action sublist identifies relevant possibilities for response; gating determines which of those possibilities are presently admissible; and action selection determines which admissible response will be executed.**

The proposed architecture therefore does not prescribe a universal action-selection algorithm. It establishes **where the action-selection problem occurs, what information reaches it, and what its output must be**. The detailed selection mechanism may subsequently be designed according to the requirements and classification of the particular robot.

Post-Action Inventory, Updating, Training, and Storage

Execution of an action does not terminate the behavioral process. The action produces consequences, and these consequences create new information concerning the robot itself, the environment, the event, and the effectiveness of the selected behavior. Before the next behavioral cycle begins, this information must be identified and incorporated into the behavioral architecture.

The first operation following an action is therefore the construction of a **Post-Action Inventory**. This inventory represents the differences between the behavioral state existing before the action and the state observed after its execution.

If the pre-action state is denoted by $\mathbf{Z^-}$, the selected action by **A***, and the observed post-action state by $\mathbf{Z^+}$, the transition may be represented as

$$\mathbf{Z^- \rightarrow A^* \rightarrow Z^+}$$

The Post-Action Inventory contains the information generated by this transition. It may include changes in robot location, event location, internal operational conditions, environmental conditions, human responses, completion or failure of the action, new observations, resource consumption, and other consequences represented within the active behavioral library. The inventory should not automatically be stored in its entirety. Different post-action information may have different behavioral significance. Some information may be temporary and relevant only to the present situation. Some may require the updating of an existing behavioral function. Other information may represent sufficiently new experience to justify training or the establishment of a new internal variable.

The post-action process may therefore be separated into three operations:

POST-ACTION INVENTORY → UPDATING / TRAINING / NEW STORAGE

Updating Existing Internal Functions

If the action produces information relevant to an already established internal function, the existing function may be updated rather than replaced.

For example, an internal function representing accumulated experience with a particular situation may change after the robot observes the consequence of its action. Similarly, an existing preference, confidence, expected consequence, human-related state, or operational quantity may acquire a new value.

Thus,

$$\textbf{Existing } I_i + \textbf{New Experience} \rightarrow \textbf{Updated } I_i$$

This updating becomes part of the robot's continuing behavioral history and consequently affects future evaluations within the domain.

$$\Omega = (X_r, X_e, T_r, T_e)$$

Training

Some post-action information may require more than an immediate state update. Repeated experience, systematic differences between expected and observed consequences, or newly acquired relationships may

justify modification of the behavioral representation through training.

Training is therefore distinguished from continuous updating. **Continuous updating maintains the present values of behavioral functions**, whereas **training may modify the underlying representation from which future behavioral functions, matching conditions, associations, or stored action possibilities are obtained.**

The exact training algorithm is not prescribed in the present paper. Depending upon the robot architecture, training may be supervised, experience based, deterministic, statistical, or implemented through another appropriate mechanism.

The important point is the location of training within the behavioral cycle:
ACTION → CONSEQUENCE → POST-ACTION INVENTORY → TRAINING → MODIFIED BEHAVIORAL LIBRARY

Training therefore does not occur in isolation from behavior. The robot's own actions and their observed consequences may become training information.

Establishing and Storing New Internal Variables

A particularly important possibility occurs when the Post-Action Inventory contains behaviorally relevant information that cannot be adequately represented by the existing internal variables.

In this case, the architecture may permit the establishment of a **new internal behavioral variable**,

$\mathbf{I_{n+1}}$

The process may be represented conceptually as
New Experience → Behavioral Significance → New Internal Variable → Storage

This operation should not imply that every new observation creates a new internal variable. Such unrestricted growth would make the behavioral library continuously expand without control. A storage mechanism is therefore required to determine whether the new information should be discarded, temporarily retained, incorporated into an existing variable, or stored as a new persistent internal representation.
The general procedure becomes
POST-ACTION INFORMATION
→ Discard
→ Temporary State
→ Update Existing I_i
→ Train Existing Representation
→ Create and Store New I_{n+1}

This introduces an important form of behavioral development. The set of internal variables need not remain permanently fixed:
$\mathbf{I_{n+1} = I_{n+1}(\Omega)}$

may become

$\mathbf{\{I_1(\Omega), I_2(\Omega), \ldots, I_n(\Omega)\} \rightarrow \{I_1(\Omega), I_2(\Omega), \ldots, I_n(\Omega), I_{n+1}(\Omega)\}}$

Consequently, the robot's future behavioral state may contain information that did not exist before the action was performed.

Updating SPARK–Action Associations

Post-action experience may also modify the relationship between a SPARK class and its stored action sublist. An action that was previously associated with a particular SPARK may prove ineffective, inappropriate, or especially useful under certain conditions. The corresponding stored association may therefore be updated, subject to the robot's authorized learning and storage mechanism.

Conceptually,
SPARK S_i + ACTION A_j + OBSERVED CONSEQUENCE → UPDATED ASSOCIATION

and, where justified,

$\mathbf{A(S_i) \rightarrow A'(S_i)}$

A new experience may also establish a previously unavailable SPARK–action association. Thus, behavioral development may modify not only the internal functions of the robot but also the action possibilities associated with future SPARKs.

This mechanism also gives a direct meaning to **robot time T_r**. Robot time represents more than the passage

of external clock time. As actions are performed and consequences are incorporated, the robot accumulates a behavioral history. Two otherwise similar events encountered at different values of $\mathbf{T_r}$ may therefore produce different internal states, different matching conditions, different SPARKs, and different Action Lists.

The complete closed behavioral cycle can now be represented as

CONTINUOUS UPDATING
Matching → Behavioral Landscape → SPARK
→ Stored Action Sublist
→ Current-State Gating
→ Active Action List
→ Action Selection
→ Action
→ Post-Action Inventory
→ Update / Train / Store
→ Modified Internal State and Library
→ Next Continuous Updating Cycle

The post-action stage therefore serves two purposes. It closes the immediate behavioral cycle by determining the consequences of the selected action, and it provides a mechanism through which those consequences can modify the robot's future behavior. **The robot does not merely act in the environment; the consequences of its own actions may become part of what the robot subsequently is.**

Development of New Internal Functions and Related Action Lists

The behavioral architecture of a robot need not remain identical throughout its operational life. Continuous updating may modify the values of existing behavioral functions, but accumulated experience and training may produce a more fundamental change: the robot may acquire **new internal behavioral functions** that were not part of its earlier behavioral state.

This distinction is important. Updating an existing internal function changes the state of an already available behavioral quantity. The development of a new internal function changes the behavioral representation available to the robot itself. Thus, learning may modify not only the numerical values of the existing state but also the **content and dimensionality of the behavioral library**.

If the internal behavioral state at robot time $\mathbf{T_r^1}$ contains

$$\mathcal{I}(T_r^1) = \{I_1(\Omega), I_2(\Omega), \ldots, I_n(\Omega)\}$$

subsequent experience and training may produce, at a later robot time $\mathbf{T_r^2}$

$$\mathcal{I}(T_r^2) = \{I_1(\Omega), I_2(\Omega), \ldots, I_n(\Omega), I_{n+1}(\Omega), \ldots, I_{n+p}(\Omega)\}$$

Here, $\mathcal{I}$ denotes the internal-function set, and $I_i(\Omega)$ denotes an individual internal behavioral function.

The robot at $\mathbf{T_r^2}$ therefore cannot necessarily be behaviorally represented by exactly the same set of internal functions that described it at $\mathbf{T_r^1}$. Its operational history has altered the behavioral architecture available for interpreting subsequent events.

The creation of a new internal function should not occur for every new observation. Much of the incoming information may be temporary, may update an existing function, or may have insufficient behavioral significance to justify permanent storage. The Post-Action Inventory introduced previously provides a location at which this distinction can be made.

A general developmental sequence may therefore be represented as

Experience / Post-Action Information
→ Evaluation
→ Update Existing Function
→ Train Existing Representation
→ Establish New Internal Function

The establishment of a new internal function, however, is only part of the required development. A behavioral concept becomes operationally useful when the robot also possesses possible responses associated with circumstances in which that function participates.

Consequently, new internal functions may require the establishment or modification of corresponding **SPARK associations and Action Sublists**.

The developmental sequence is therefore extended to

NEW EXPERIENCE
→ New Internal Function
→ New or Modified Matching Relationships
→ New or Modified SPARK Associations

→ New or Modified Action Sublists

Consider a robot that initially possesses no behavioral representation associated with playing tennis. At this stage, observing a tennis court need not produce a tennis-related behavioral possibility because the corresponding internal representation and behavioral associations may not exist within its library.

Following appropriate experience or training, however, the robot may acquire tennis-related internal functions. These may subsequently participate in matching with relevant external functions. Tennis-related circumstances can then contribute to a SPARK that could not previously have occurred in the same behavioral form.

The newly available SPARK must also have access to an appropriate set of possible actions. Its associated stored Action Sublist might include observing, approaching the court, identifying a partner, requesting participation when appropriate, playing, postponing the activity, or abandoning the possibility.

Thus, learning does not merely change the probability, weight, or numerical value associated with an old action. It may change **what the robot is capable of behaviorally considering**.

This distinction can be represented as
Old Behavioral Library
{Internal Functions} + {SPARK Associations} + {Action Sublists}

followed by experience and training,
↓
Expanded Behavioral Library
{Existing Internal Functions + New Internal Functions}
+ {Existing + New SPARK Associations}
+ {Existing + New Action Sublists}

This developmental process gives additional significance to robot time $\mathbf{T_r}$. Robot time is not merely elapsed clock time. It also identifies the accumulated behavioral history of the robot. Two observations of the same external event at different robot times may therefore interact with different internal-function sets and different stored behavioral possibilities.

Consequently,
same external event + different T_r
may produce
different matching → different SPARK → different Action List → different behavior
The robot at a later robot time is therefore not necessarily behaviorally equivalent to its earlier version. It may possess knowledge, internal functions, SPARK relationships, and available actions that previously did not exist.

This leads to an important principle of the proposed dynamic behavioral architecture:
Learning may change not only the values of the robot's behavioral functions, but also the set of behavioral possibilities available to the robot.

The architecture is therefore dynamic at two different levels. At the first level, existing functions are continuously updated as the robot and environment change. At the second level, experience and training may modify the **structure of the behavioral library itself** by introducing new internal functions and their related SPARK and action associations.

The complete developmental loop can consequently be written as
ACTION → POST-ACTION INVENTORY → EXPERIENCE / TRAINING → UPDATED OR NEW INTERNAL FUNCTIONS → UPDATED MATCHING STRUCTURE → UPDATED SPARK ASSOCIATIONS → UPDATED ACTION SUBLISTS → MODIFIED BEHAVIORAL CAPACITY → NEXT BEHAVIORAL CYCLE

In this sense, the robot does not merely accumulate data. **Its behavioral architecture may develop with experience.**

Conclusion

This paper has introduced **SPARK and the Dynamic Behavioral Architecture** as a framework for examining a fundamental question in autonomous robotics: how can a behavioral initiative or candidate goal emerge when it has not been explicitly supplied as the immediate objective of the robot?

The proposed formulation distinguishes this problem from conventional autonomous execution. An

autonomous robot may possess substantial freedom in determining how to accomplish an externally imposed mission while the general purpose of its behavior remains predefined. The combat soldier robot illustrates this case. In contrast, the tennis robot illustrates a different possibility: circumstances, accumulated experience, internal state, and external events may interact so that *playing tennis* emerges as a candidate behavioral direction even though it was not supplied as the current goal. The distinction is therefore not simply between autonomous and non-autonomous systems, but between **autonomous execution of an existing purpose and the emergence of a new behavioral initiative**.

The architecture was formulated over a four-dimensional agent–event domain containing robot location, event location, robot time, and event time. Internal and external behavioral functions may vary throughout this domain. At a particular present robot state, however, robot location and robot time may be fixed, reducing the instantaneous event representation to a conditional two-dimensional landscape. The resulting behavioral landscape may contain valleys, peaks, saddles, or other local structures, but the existence of such structures does not imply that behavioral determination must be formulated as an optimization problem.

Robot classification provides an additional reduction. A robot is not required to contain or continuously evaluate every conceivable behavioral interaction. Its classification determines its library capacity, and its active library determines which internal and external behavioral functions participate in the current state. The architecture can therefore remain open while individual implementations remain bounded.

Within this framework, externally originating information may become part of the robot's internal behavioral state. A master's command provides an important example. Once received, interpreted, and retained, the command becomes one of the internal behavioral functions. It may possess considerable authority, but it does not constitute the complete behavioral mechanism. Command, accumulated experience, internal needs, learned behavioral quantities, and current external conditions may coexist within the continuously updated behavioral state.

Continuous updating provides the dynamic connection between the robot and its changing environment. Matching may occur through behavioral-function values, their first derivatives, their second derivatives, or combinations of these quantities. A SPARK may emerge when these relationships become behaviorally significant. However, SPARK was deliberately separated from final decision and action. It identifies the emergence of a behavioral possibility rather than prescribing its ultimate consequence.

An important extension of this distinction is the introduction of the **Action List**. Following a SPARK, the robot does not necessarily search its complete universe of possible actions, nor does the SPARK directly prescribe a single response. Instead, identifiable SPARK classes may be associated with stored Action Sublists. The relevant sublist is retrieved and gated according to the current state, classification, authority, physical capability, and operational constraints. The resulting Active Action List then becomes the input to a separate action-selection mechanism.

The post-SPARK sequence is therefore
SPARK → STORED ACTION SUBLIST → CURRENT-STATE GATING → ACTIVE ACTION LIST → ACTION SELECTION → ACTION

No universal action-selection algorithm has been imposed. Optimization, deterministic comparison, elimination, priority rules, learned procedures, or other mechanisms may eventually be appropriate for different implementations. The architecture identifies the location and inputs of the selection problem without assuming its solution in advance.

Execution also does not terminate the behavioral process. The consequences of an action generate a **Post-Action Inventory** containing changes in the robot, the environment, the event, and the observed outcome of the action. This information may be discarded, temporarily retained, used to update existing internal functions, incorporated through training, or stored as new behavioral information.

A particularly important consequence is that learning need not merely modify the values of existing internal functions. Experience and training may lead to the

establishment of **new internal behavioral functions**, new or modified matching relationships, new SPARK associations, and corresponding new Action Sublists. The behavioral capacity of the robot may therefore develop during its operational history.

Consequently, the robot at a later robot time $\mathbf{T_r^2}$ need not be behaviorally equivalent to the same robot at an earlier robot time $\mathbf{T_r^1}$. The difference may involve more than changed numerical states. The later robot may possess behavioral functions, associations, and possible actions that did not previously exist.

The complete architecture can therefore be summarized as

Classification
→ Library Capacity
→ Active Library
→ Internal And External Functions
→ Continuous Updating
→ Dynamic Matching
→ Behavioral Landscape
→ SPARK
→ Stored Action Sublist
→ Active Action List
→ Action Selection
→ Action
→ Post-Action Inventory
→ Updating / Training / Storage
→ New or Modified Internal Functions
→ New or Modified SPARK–Action Associations
→ Next Behavioral Cycle

The exact mechanism producing SPARK has intentionally not been forced into a predetermined mathematical form. In particular, the four-dimensional formulation does not imply optimization, and the presence of derivatives does not require minimization or maximization. These remain possible mechanisms rather than assumptions. Similarly, the detailed formulation of human–robot interaction, training procedures, and behavioral functions for every robot classification remains outside the present scope.
The principal objective of the paper is therefore not to claim a complete solution to autonomous goal generation. It is to establish an engineering architecture in which the problem can be stated more precisely: **behavioral functions change, their relationships change, SPARKs may emerge, possible actions become available, actions produce consequences, and those consequences may change the behavioral architecture that will confront the next event.**

In this sense, the robot does not merely execute behavior within a changing world. T**hrough experience, the robot itself may become behaviorally different from the robot that entered the previous cycle.**

References

1. Brooks, R. A. (1991) Intelligence without representation. Artificial Intelligence 47: 139-159.
2. Siciliano, B. and Khatib, O. (2008) Springer Handbook of Robotics. Springer. https://link.springer.com/book/10.1007/978-3-540-30301-5.
3. Oudeyer, P. Y. and Kaplan, F. (2007) What is intrinsic motivation? A typology of computational approaches. Frontiers in Neurorobotics 1: 6.
4. Baranes, A. and Oudeyer, P. Y. (2013) Active learning of inverse models with intrinsically motivated goal exploration in robots. Robotics and Autonomous Systems 61: 49-73.
5. Moulin-Frier, C., Nguyen, Sao M. and Oudeyer, P. Y. (2014) Self-organization of early vocal development in infants and machines: the role of intrinsic motivation. Frontiers in Psychology 4: 1006.
6. Colas, C., Karch, T., Sigaud, O. and Oudeyer, P. Y. (2022) Autotelic agents with intrinsically motivated goal-conditioned reinforcement learning: a short survey. Journal of Artificial Intelligence Research 74: 1159-1199.
7. Merrick, K. E. (2017) Value systems for developmental cognitive robotics: a survey. Cognitive Systems Research 41: 38-55.