Robot Skill Puzzles: Decentralized Skill Priors for Humanoid–Environment Interaction

Zhiyang Dou[1*], Jintao Lu[1*], Boyang Yu[1*], Zeyu Cao[2], Cheng Lin[3], Yuan Liu[4], Taku Komura[1]
[1]The University of Hong Kong [2]University of Cambridge [3]Macau University of Science and Technology
[4]The Hong Kong University of Science and Technology [*]Equal contribution

***Abstract*—Behavior foundation models are becoming a practical substrate for generalist robot control, yet they still face a misaligned scaling target: a single resident policy is expected to internalize the geometry, articulation, force limits, affordance frames, and safe modes of use for a growing long tail of objects. We propose *Robot Skill Puzzles* (RSPs), a decentralized alternative in which an object exposes a signed, typed action artifact containing articulated assets, reference kinematics, advisory constraints, and optionally a compact object-specific controller. The robot verifies this artifact, maps its task-space interface to its own morphology, and executes it only through a robot-owned safety envelope. This keeps actuator authority on the robot while moving object-local interaction priors to the source best positioned to specify and maintain them. We formalize RSPs as object-resident contracts, state validity and safety conditions, and give a capacity argument separating resident controller footprint from object-catalog size. Two preliminary studies probe the factorization rather than validate a deployed system. In a synthetic scaling study over 1000 action skills, a fixed-capacity centralized controller reaches 0.54 normalized error at matched resident footprint, whereas a decentralized backbone-plus-action-head scheme remains near 0.14. The construction favors decentralization by design, so we read it as a capacity intuition rather than a benchmark result. In a preliminary experiment with physically simulated characters, warm-starting object-specific controllers from a shared prior lowers early whole-body tracking error on five held-out objects under a matched fine-tuning budget, which suggests that the object-hosted controller ingredient is producible. We present RSPs as an interface and authority-separation proposal for behavior foundation models, position them as verified object-local priors for planning, semantic grounding, whole-body execution, and long-horizon recovery, and scope end-to-end execution together with latency and safety-benefit measurement as the next step. The RSP/1.0 interaction protocol is specified in full in the appendix.**

I. Introduction

Large behavior foundation models are emerging as the dominant route to general-purpose robot control. RT-2, RT-X, Octo, OpenVLA, and π_0 amortize broad manipulation knowledge across large robot datasets and Internet-scale vision-language pretraining [38, 27, 9, 19, 4]. Large embodied models such as PaLM-E make the same point at the planning and reasoning level [7]. Humanoid behavior foundation models pursue an analogous goal for whole-body systems: learn a reusable behavioral prior once, then adapt it to new control modes, embodiments, and tasks [36, 37, 12, 8, 29, 14, 13, 32].

This line of work expands what a single robot-side policy can do, but it leaves a scaling question: what still has to live inside that resident policy? Robots deployed in homes, workplaces, hospitals, and public spaces encounter a long tail of interactive objects. Useful behavior depends not only on recognizing an object, but also on knowing its articulated geometry, affordance frames, force and velocity limits, state predicates, failure modes, and safe modes of use. Encoding all of this object-local knowledge inside one behavior model is attractive when the object set is fixed; it becomes brittle as the catalog grows and many resident skills are irrelevant to the current scene.

Recent robot-control systems show how far agent-side learning can go. Teleoperation and imitation systems distill whole-body human motion into deployable humanoid policies, neural whole-body controllers unify multiple control contexts, and physics-based human–object interaction methods learn dynamic interaction skills from demonstration [12, 8, 14, 35]. These systems strengthen the case for behavior foundation models, but they also make the remaining bottleneck sharper: where should the object-specific part of interaction knowledge reside?

Robot Skill Puzzles take a systems position (Fig. 1). The behavior foundation model remains the robot's resident prior for perception, language, task sequencing, retargeting, and whole-body execution. The object supplies its local interaction contract (Fig. 2 shows a kitchen-like scene, with the object-side carrier in blue publishing the contract and the robot keeping whole-body authority). A cabinet publishes its hinge, handle frame, safe swing region, terminal predicate, and recovery states. A tray publishes its support polygon, tilt limit, payload state, and two-hand constraints. A deformable strap publishes a parameterized compliance model and yield limit. The robot verifies the package, tests whether the contract is feasible for its body and safety envelope, and executes with its own controller.

RSPs do not transfer control authority to objects. They separate object-local interaction priors from robot-side planning and safety. An RSP gives the object a machine-checkable way to state what it knows about itself, while the robot retains actuator authority, feasibility checking, and runtime monitoring. This recasts part of open-world interaction as an interface problem rather than a single-model scaling problem: which physical contracts should be resident, which should be inferred online, and which should be published by the object

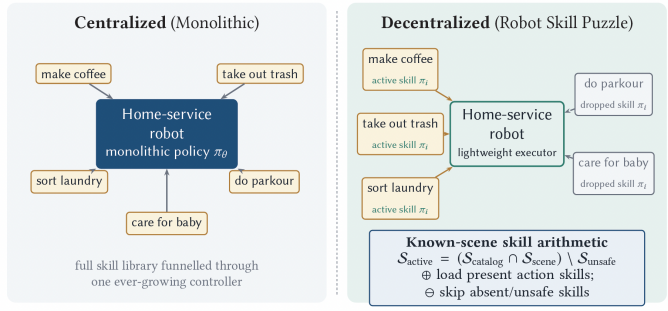

Fig. 1. Centralized and decentralized views of humanoid–environment interaction. In the RSP view, the robot loads only present, verified, and safe object-side action skills instead of carrying an ever-growing resident skill library.

Fig. 2. Robot Skill Puzzles in a kitchen-like scene. Blue denotes the tool or object-side interaction carrier; yellow denotes the target object when present. The robot keeps whole-body control authority, while the blue tool or carrier provides the object-local interaction contract that can be verified and bound before contact.

that owns the mechanism?

Contributions. We (i) define the Robot Skill Puzzle as an object-resident unit of interaction and position it as an interface and authority-separation proposal rather than a finished learning system; (ii) formalize it through a typed interface, a robot-owned safety envelope, and a capacity argument separating resident footprint from object-catalog size; (iii) report two preliminary studies, a synthetic scaling study over 1000 action skills and a fine-tuning study on held-out interaction objects, that probe the factorization without yet exercising object-published verification, binding, and safety filtering end-to-end; and (iv) specify the RSP/1.0 handshake, together with an account of how object contracts are authored, versioned, and revoked, as a concrete interface for signed object–robot interaction artifacts.

II. Robot Skill Puzzles

An RSP is a signed interaction artifact exposed by an object over a local communication channel or retrieved from a trusted registry after object identification. The artifact is declarative at the boundary. Executable policies may be included, but they are optional accelerators; the authoritative content is the typed contract that the robot can inspect before contact.

TABLE I
Core fields in a Robot Skill Puzzle. Fields are object-local and task-space so different robots can bind them to their own embodiment.

Field	Object-side content	Robot-side use
Identity	UUID, version, signature, expiry	Trust and rollback checks
Geometry	Mesh, links, joint axes, limits	Registration and collision tests
Affordances	Grasp, push, pull, brace, support frames	Grounded primitive binding
Constraints	Force, torque, speed, keep-out zones	Feasibility and safety filters
Goal state	Predicates, latch/status telemetry	Progress monitoring
Recovery	Retry, release, retreat, request assistance	Stage-local fallback
Policy	Optional sandboxed controller	Contact-rich residual skill

Table I summarizes the fields, and Fig. 3 shows the resulting handshake: the object broadcasts, the robot verifies, binds the contract to its morphology, and executes under its own safety envelope.

Concretely, a puzzle for object o is a tuple $P_o = \langle A, \mathcal{K}, C, \pi, I \rangle$. Here A is an articulated asset containing geometry, joints, inertials, and collision structure; $\mathcal{K}$ is a set of admissible task-space reference trajectories; C is an advisory constraint set containing keep-out regions, force bounds, velocity limits, and state predicates; π is an optional object-specific controller; and I is the typed interface declaring observation/action spaces, units, reference frames, controller flavor, and control rate. The load-bearing design choice is that $(\mathcal{K}, C, \pi)$ are expressed in object-local task space, not in any particular robot's joint space.

The contract is therefore expressed in task space rather than joint space. A drawer publishes a pull axis and force envelope, not a shoulder trajectory. A door publishes a hinge frame and safe swing region, not the robot's footstep plan. A long tool publishes a leading-hand frame, support-hand frame, and torque bound, not the arm allocation. This separation is what makes the same object contract usable by a mobile manipulator, a tabletop bimanual system, or a humanoid. The robot decides how to allocate base, torso, arms, hands, compliance, and null-space posture.

The execution invariant is robot authority over object advice. Before importing any executable asset, the robot verifies the signature, checks version and expiry, parses the typed fields, and tests the contract against its reachability, collision, torque, balance, and human-proximity constraints. The executed admissible set is the intersection of the robot's certified safe set and the object's advisory envelope. An object can tighten the robot's envelope but cannot relax it.

The same rule gives a deterministic skill-selection step. If $\mathcal{S}_{\text{catalog}}$ is the set of discovered object-advertised skills and $\mathcal{S}_{\text{scene}}$ is the set present in the local scene, the robot forms

$$\mathcal{S}_{\text{active}} = (\mathcal{S}_{\text{catalog}} \cap \mathcal{S}_{\text{scene}}) \setminus \mathcal{S}_{\text{unsafe}}.$$

The unsafe set is computed by the robot from revocation

status, morphology mismatch, unsatisfied dependencies, declared conflicts, memory and latency budgets, and an empty intersection between the object constraints and the robot-owned safety envelope. Thus a behavior foundation model can ask for *open drawer* or *lift tray*; the executor loads only the verified object contracts that are present, compatible, and safe.

A. Authoring, Versioning, and Revocation

An interface is only as useful as the contracts written for it, so the natural question is who produces an RSP and how it stays correct. The proposal does not assume a single author. A manufacturer or venue operator can author a contract directly from the design data it already owns, exporting geometry and joints from CAD to URDF, MJCF, or SDFormat [28, 33, 25] and declaring the affordance frames, force envelopes, and terminal predicates that define correct use. Where design data is incomplete, the same fields can be fitted in simulation or distilled from teleoperated demonstrations, and an optional object-specific controller can be trained the same way. A third party can also certify a contract without re-authoring it, since verification and provenance are separated from authorship by signed metadata [30, 34, 26]. In every case the object publishes a claim, and the robot still decides whether that claim is admissible for its body and safety envelope.

This shifts part of the long-tail burden from robot-side policy capacity to object-side contract production, which is a real cost rather than a free lunch. We argue the trade is favorable because the party that manufactures or maintains a mechanism already holds its most precise physical description, and because the same contract, authored once, is reusable by every robot that can bind the interface. Each contract carries a semantic version and expiry, and a physical change, recall, or discovered error is handled by publishing a new version and revoking the old one through the trust root rather than editing a value in place. A robot resolves the current version from signed freshness metadata rather than a local highest-seen cache, and a stale but validly signed contract is treated as suspect until runtime monitoring confirms it against observed motion and force.

B. Relation to Adjacent Interfaces

Table II places RSPs against adjacent interfaces. Affordance templates and articulated-object resources already expose object-local structure, and robot-side skill libraries already package reusable behavior, but none treats the interaction contract as an object-authored artifact that is signed, versioned, revocable, sandboxed for optional executable content, and auditable at teardown while the robot keeps sole actuator authority. That combination, rather than any single field, is what RSPs contribute.

TABLE II
WHERE RSPS DIFFER FROM ADJACENT INTERFACES. PRIOR LINES CONTRIBUTE OBJECT-LOCAL STRUCTURE; RSPS ADD OBJECT-SIDE AUTHORSHIP AS A SIGNED, REVOCABLE DEPLOYMENT ARTIFACT EXECUTED UNDER ROBOT-OWNED SAFETY AUTHORITY.

Property	Afford. tmpl.	Artic. KB	Skill lib.	RSP
Object-side authorship	part.	part.	no	yes
Signed, versioned contract	no	no	no	yes
Typed execution interface	part.	no	part.	yes
Revocation and freshness	no	no	no	yes
Sandboxed optional controller	no	no	part.	yes
Auditable transcript	no	no	no	yes
Robot-owned safety authority	part.	n/a	part.	yes

III. WHY THIS FITS BEHAVIOR FOUNDATION MODELS

RSPs are meant to complement behavior foundation models, not compete with them. Behavior foundation models provide what an object cannot: open-vocabulary language understanding, scene-level reasoning, cross-object sequencing, visual grounding, and broad embodied priors. RSPs provide what a resident model should not have to memorize: the object-local physical contract. The combination gives a clearer division of labor than a monolithic agent-centric policy.

Planning. A foundation-model planner can propose a plan skeleton from language and perception, then ground each sub-goal by matching it against verified object contracts. The planner no longer has to infer from appearance alone whether a cabinet affords `pull` or `push`; the affordance set is closed by the object's declared fields. Preconditions and effects convert vague sub-goals into typed predicates. If no verified object in the scene declares an affordance that can establish a desired predicate, the plan is rejected as ungrounded before contact. This is particularly relevant to current questions about how foundation models should interact with symbolic, formal, and tool-like planning interfaces.

Whole-body and bimanual execution. For humanoids and bimanual robots, the object-level plan is rarely sufficient. A two-hand pull may be feasible for the end effectors but unsafe for balance; a tray lift may satisfy grasp poses while violating a tilt limit; a door opening may require a retreat path. RSPs expose the terms a whole-body controller needs: coupled hand constraints, support regions, body-placement hints, wrench bounds, and telemetry for progress. The controller remains robot-specific, but the object contract makes the optimization target explicit.

Articulated and deformable objects. The case is strongest when the relevant facts are difficult to infer from a single view. Articulated-object resources and affordance-conditioned policies show the value of explicit intermediate representations [10, 11, 3, 21, 22]. RSPs move that representation from an offline dataset or robot-side estimate into the object at deployment time. A typed hinge axis or yield limit can be checked before a learned policy acts; a latent memory of a hinge buried in a large model cannot.

Semantics and trust. RSPs also provide a concrete attachment point for specification. Formal methods and trustworthiness work emphasize that autonomous systems need explicit

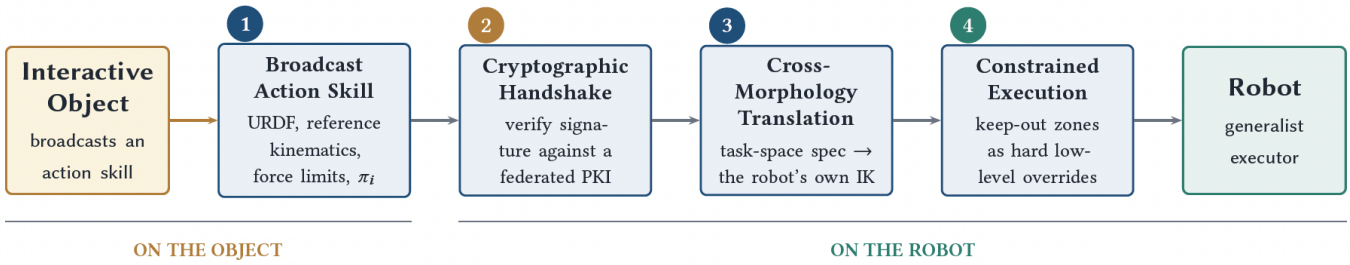

Fig. 3. The Digital Instruction Manual Protocol. The object broadcasts an action skill packaged as a Robot Skill Puzzle; the robot verifies, translates, and executes it under hard safety constraints.

statements of the assumptions under which behavior is safe [20, 1]. An object-published contract makes those assumptions local, versioned, and auditable. It also turns the robot–object stack into a distributed system in the practical sense: identity, versioning, revocation, stale state, and partial failure become first-class concerns rather than afterthoughts [6].

IV. Formal Model and Evaluation

Scope of the evidence. The experiments below evaluate two components of the framework in isolation: the capacity separation between resident footprint and catalog size, and warm-start transfer of object-specific controllers. They do not instantiate the full signed, verified, safety-mediated pipeline; no RSP artifact is published, verified, and executed end to end in this paper. Sections VI and VIII state this boundary explicitly.

The formal model has one central rule: the object proposes interaction structure, and the robot retains execution authority. Let E_r denote the robot's certified safe set and let $\Phi_r^{-1}(C)$ denote the robot-frame pre-image of the object's advisory constraint set. Under puzzle P_o on robot r, the executed admissible set is

$$S_o^r = E_r \cap \Phi_r^{-1}(C).$$

This set-intersection property is an architectural choice. It means the object can only tighten the robot's admissible set; it cannot enlarge the robot's certified envelope or write actuator commands. Trajectory-level safety still depends on the robot-side runtime-assurance layer, such as a control-barrier-function or Simplex-style filter [2, 31].

A puzzle is valid for a robot only when its references are reachable and its safe-and-reachable region is nonempty. This makes cross-morphology use a reachability and inverse-kinematics query rather than a physical rollout: the same object-local contract may be accepted by a humanoid, rejected by a tabletop arm, and accepted with different footstep or hand-allocation choices by a mobile manipulator. For behavior foundation models, this is the useful boundary. The model proposes and sequences goals; the RSP verifier decides which object-local contracts are admissible before contact.

The scaling claim is about resident footprint. A centralized behavior model must carry a skill-conditioned map broad enough for the whole object catalog. An RSP system carries

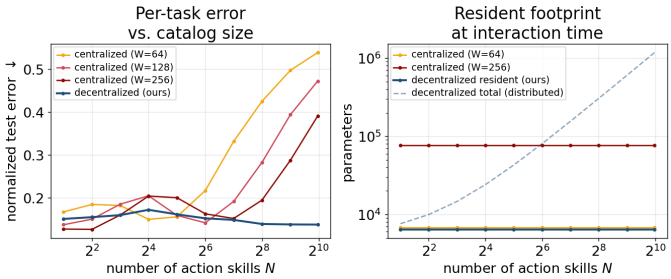

Fig. 4. Scaling study: decentralization versus a centralized controller as the action-skill catalog grows to N=1000. The decentralized scheme keeps resident interaction-control footprint constant by loading only the current object-specific head.

a shared robot-side prior and loads only the relevant object-specific head at interaction time. In the scaling study, each action skill is a random task with a shared interaction structure plus a skill-specific residual. At N=1000 skills, the matched-footprint centralized model reaches 0.539 normalized MSE, while the decentralized backbone-plus-head scheme remains at 0.137 (Fig. 4); even a 12× larger centralized model remains 2.9× worse. The crossover occurs near $N \approx 128$, so the claim is not that decentralization dominates at small catalog size. It is that open-ended object catalogs create resident-model pressure that object-local storage can absorb.

In a known environment, the robot rarely needs the full catalog. If a scene contains only K of N available skills, a selector activates only the present, verified, and safe heads (Fig. 5). At N=1000 and K=10, the active footprint is roughly 45× smaller than the full-library footprint, with no loss when the gate selects the correct heads. The failure mode is explicit: routing to a wrong expert or dropping a needed expert increases error approximately in proportion to the gate error rate (measured at N=200; Fig. 7).

The second experiment asks whether the demanding component, an object-specific controller, can be produced for contact-rich interaction. We run a preliminary experiment with physically simulated characters, built on a physics-based character–object interaction framework [35]. A controller trained on one interaction subject serves as the shared prior, then is fine-tuned on five objects held out from that prior: a large box, plastic box, suitcase, small box, and trash can. Warm-start and from-scratch controllers use the same short fine-tuning budget,

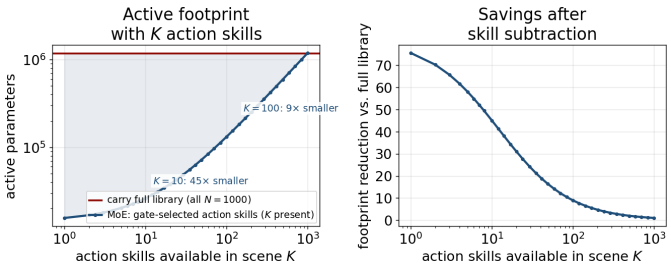

Fig. 5. Known-scene activation. Active parameters scale with the local scene size K, not the global catalog size N, when the present object-hosted heads are selected correctly.

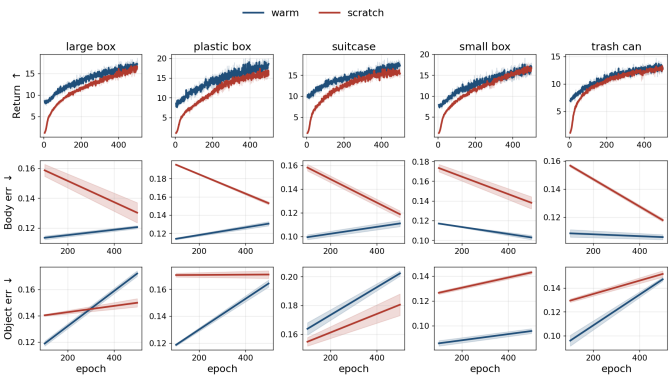

Fig. 6. Fine-tuning object-specific controllers from a shared prior on five held-out interaction objects. The warm-started controller has a consistent early advantage in whole-body tracking under the same short budget.

learning rate, reward, and evaluation protocol. At epoch 100, warm-start reduces body-tracking error by 29–42% across all objects (Fig. 6). At the common epoch-500 evaluation point, warm-start remains lower on body error for every object, and reaches the from-scratch final return 1.2–2.1× sooner (Fig. 8).

The central benchmark should hold the behavior foundation model and robot controller fixed while varying the source of object knowledge, so that the comparison isolates where the knowledge lives rather than which policy is stronger. A *vision-only* condition infers affordances, articulation, and limits from observations and language. A *documentation* condition also retrieves a static manual, CAD file, or description, but must parse and register it; this is the robot-side retrieval and modular-skill-library baseline that the proposal must beat. An *object-published* condition broadcasts a verified declarative RSP already grounded to geometry and state, and a final condition adds the optional object-hosted controller. Metrics should include task success, hallucinated-affordance rate, pre-contact infeasibility detection, unsafe contact, stage recovery, gate error, first-contact latency, and out-of-distribution degradation.

V. Empirical Details

The scaling study uses N action skills, each a fixed random task with a shared interaction structure T and a skill-specific residual ρ_o drawn independently per skill, so the target is $g_o(x) = T(x) + \rho_o(x)$. The centralized baseline

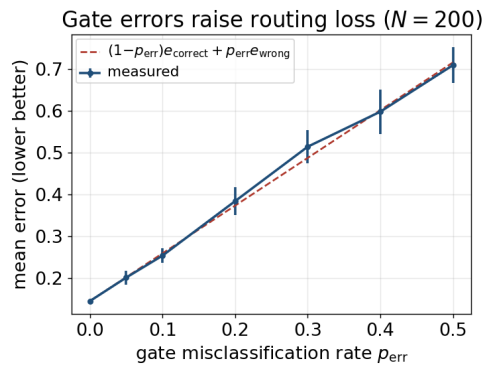

Fig. 7. Gate sensitivity in the MoE study (N=200). Mean per-task error rises approximately linearly with injected skill-misclassification rate, from 0.14 under a correct gate to 0.71 at p_{err}=0.5.

is a skill-conditioned MLP; the decentralized model is a shared backbone plus object-specific action heads, only one of which is loaded at interaction time. Capacity is matched by resident footprint: the centralized width is set so that its resident parameter count equals the decentralized backbone plus one head, and we additionally report an over-provisioned centralized model at 12× that footprint. Results are averaged over random seeds for the task draw and initialization, and we report trends rather than significance tests because the study is a controlled illustration of interference. The construction is deliberately favorable to decentralization, since the target splits cleanly into a shared term and independent residuals; the honest reading is that open-ended catalogs create resident-model pressure, not that any centralized architecture must fail. At small catalog sizes ($N \leq 100$), a single model is competitive and often preferable, and the decentralized scheme overtakes the best centralized configuration around $N \approx 128$.

The known-scene experiment measures active footprint at interaction time rather than total offline storage. This distinction is central to RSPs. A deployed robot may know that a large catalog exists, but a specific kitchen scene exposes only a small subset of handles, trays, bins, and tools. The selector therefore loads only the verified heads for objects present in the scene and leaves absent or unsafe heads inactive. Gate errors are modeled separately because they are a different failure mode: the resident footprint can still be small, but routing a goal to the wrong object-side prior increases per-task error roughly in proportion to the misclassification rate (Fig. 7).

The physics study is intentionally small: one shared prior, five held-out objects, three fine-tuning seeds, and a short 500-epoch budget. Table III reports the common evaluation checkpoint. Warm-start gives lower body-tracking error on every object. Object-pose error is mixed: it is lower for warm-start on three objects but higher on the large box and the suitcase, the two bulkiest items, so we do not claim a net object-pose gain. We read this as the shared prior prioritizing whole-body stability, which keeps the body on the reference at the cost of a slightly looser object pose on the heavy items that perturb balance most. The metric is coupled to episode

TABLE III
EPOCH-500 FINE-TUNING RESULTS WITH PHYSICALLY SIMULATED CHARACTERS. LOWER IS BETTER.

Object	Body err. warm	Body err. scratch	Object err. warm	Object err. scratch
Large box	**0.121**	0.130	0.172	**0.150**
Plastic box	**0.131**	0.153	**0.164**	0.171
Suitcase	**0.111**	0.119	0.202	**0.180**
Small box	**0.103**	0.138	**0.096**	0.143
Trash can	**0.106**	0.118	**0.147**	0.152

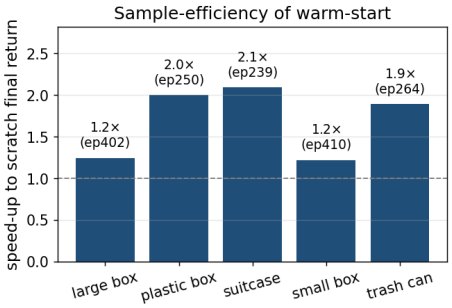

Fig. 8. Warm-start reaches the from-scratch final return 1.2–2.1× sooner across the five held-out objects.

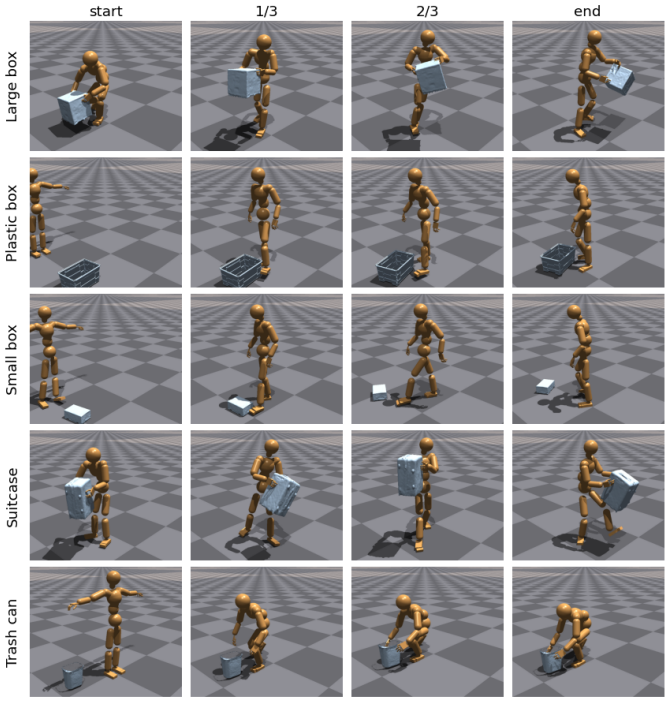

Fig. 9. Rollouts of warm-started object-specific controllers on held-out interaction objects. The strip complements the tracking curves by showing the learned whole-body approach, contact, and object manipulation phases.

survival in the underlying tracker rather than an independent accuracy measure, so we report it as a secondary diagnostic.

The fine-tuning study is meant as evidence for reusable object-conditioned initialization, not as a claim of final-performance saturation. The shared prior gives the controller a contact-rich whole-body strategy before seeing the held-out object. Fine-tuning then adapts the residual object-specific details: where to place the hands, how much body lean is needed before contact, and how to stabilize the object once motion begins (Fig. 9). This is exactly the role an RSP prior should play in the larger system: reduce early interaction cost while leaving the robot-side controller responsible for balance, contact safety, and recovery.

VI. COVERAGE ACROSS INTERACTION REGIMES

The same interface should be stressed in regimes that usually appear as separate robot-learning problems. Table IV lists five cases that we use as design targets. None of them is evaluated here; they are the coverage requirements that define the benchmark we intend to build. The common question is whether a robot can bind a high-level foundation-model plan to a physical contract that is explicit enough for verification and still abstract enough for different bodies.

These cases expose different failure boundaries. In long-horizon manipulation, the relevant quantity is not only final success but whether the robot can stop at the first ungrounded stage before compounding errors. In bimanual and whole-body tasks, success depends on whether the object contract gives the controller the right coupled constraints rather than a single end-effector target. For deformable and articulated objects, the value is early rejection of contracts that are inconsistent with observed compliance or joint motion. For semantic reasoning, the useful signal is not a richer natural-language description; it is a typed set of predicates and contraindications that a planner can audit. This makes RSPs a benchmark for interface design as much as for policy learning.

TABLE IV
COVERAGE CASES FOR A UNIFIED RSP BENCHMARK. EACH CASE KEEPS THE ROBOT-SIDE BEHAVIOR MODEL FIXED AND CHANGES ONLY THE OBJECT-PUBLISHED CONTRACT. THESE ARE PROPOSED EVALUATION TARGETS; NONE IS EVALUATED IN THIS PAPER.

Case	Object-published contract	Robot-side stress test
Long-horizon cabinet	Latch, hinge, terminal predicate, retry states	Stage progress, failure recovery, stale-state rejection
Bimanual tray	Support polygon, tilt limit, paired hand frames	Coupled hands, balance margin, payload monitoring
Deformable strap	Anchor frames, compliance range, yield limit	Contact adaptation without unsafe over-tension
Tool use	Lead/support roles, torque bound, active tip frame	Role assignment across morphology and viewpoint
Semantic goal	Predicate vocabulary, affordance set, contraindications	Grounded planning without hallucinated actions

VII. ARTIFACT AND REPRODUCIBILITY PLAN

The open-source release accompanying this paper is organized around three artifacts. First, the RSP/1.0 schema and reference verifier: the typed manifest format of Appendix A with a parser, signature check, and the deterministic skill-selection rule of Sec. II, so that others can author and validate contracts for their own objects. Second, the scaling and

known-scene studies: task generators, model configurations, and evaluation scripts for the centralized, decentralized, and gated variants, with the seeds used here, so the capacity results are reproducible from a single command. Third, the character fine-tuning protocol: the shared-prior checkpoint recipe, the five held-out object configurations, and the common 500-epoch evaluation schedule behind Table III. The end-to-end demonstration still missing from the evidence, one object publishing a signed RSP that a robot verifies, binds, and executes, will be built on the same released stack.

VIII. Limitations

RSPs do not remove the need for robot learning. A robot still needs robust perception, registration, planning, control, and safety filtering. The interface also creates new systems problems: object identity, certificate management, state freshness, revocation, privacy, and denial of service. A stale but valid contract can be dangerous if the object's physical state changed after publication; runtime monitoring must therefore validate the contract against observed motion and force. The safety intersection $S_o^r = E_r \cap \Phi_r^{-1}(C)$ is stated rather than validated: computing the certified safe set E_r for a real robot, enforcing the intersection during contact, and propagating registration uncertainty into it remain open, so the design should be read as safety-aware rather than safety-proven. The empirical studies test capacity separation and warm-start transfer in isolation; an end-to-end demonstration in which one object publishes an RSP that a robot verifies, binds, and executes is the immediate next measurement. Beyond these, the interface adds engineering surface: debugging a foundation-model stack together with object contracts and their communication protocol is harder than debugging either side alone; each object-specific controller still needs training or adaptation; and producing packages for legacy or passive objects is unaddressed. Finally, some objects will not publish packages at all. The robot must retain conservative fallback behavior for passive, unknown, or untrusted objects.

IX. Conclusion

Behavior foundation models make robot control more general, but they should not serve as the sole repository for every object-specific physical convention in the world. Robot Skill Puzzles offer a complementary factorization. The robot keeps the general behavior prior and all safety authority; the object publishes the local interaction contract that only it can state reliably. This gives planners, whole-body controllers, and semantic reasoners a verified substrate on which to compose behavior, and it gives the community a concrete benchmark question: which parts of open-world manipulation should be learned by the agent, and which should be published by the world?

References

[1] Dhaminda B. Abeywickrama, Amel Bennaceur, Greg Chance, Yiannis Demiris, Anastasia Kordoni, Mark Levine, Luke Moffat, Luc Moreau, Mohammad Reza Mousavi, Bashar Nuseibeh, Subramanian Ramamoorthy, Jan Oliver Ringert, James Wilson, Shane Windsor, and Kerstin Eder. On specifying for trustworthiness. *Communications of the ACM*, 67(1):98–109, 2024.

[2] Aaron D. Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European Control Conference (ECC)*, pages 3420–3431, 2019. doi: 10.23919/ECC.2019.8796030.

[3] Shikhar Bahl, Russell Mendonca, Lili Chen, Unnat Jain, and Deepak Pathak. Affordances from human videos as a versatile representation for robotics. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13778–13790, 2023.

[4] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. In *Robotics: Science and Systems (RSS)*, 2025. arXiv:2410.24164.

[5] Daniel Camps-Mur, Andres Garcia-Saavedra, and Pablo Serrano. Device-to-device communications with WiFi direct: Overview and experimentation. *IEEE Wireless Communications*, 20(3):96–104, 2013.

[6] Mark Cavage. There is no getting around it: You are building a distributed system. *Communications of the ACM*, 56(6):63–70, 2013.

[7] Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, Wenlong Huang, Yevgen Chebotar, Pierre Sermanet, Daniel Duckworth, Sergey Levine, Vincent Vanhoucke, Karol Hausman, Marc Toussaint, Klaus Greff, Andy Zeng, Igor Mordatch, and Pete Florence. PaLM-E: An embodied multimodal language model. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, volume 202 of *Proceedings of Machine Learning Research*, pages 8469–8488, 2023.

[8] Zipeng Fu, Qingqing Zhao, Qi Wu, Gordon Wetzstein, and Chelsea Finn. HumanPlus: Humanoid shadowing and imitation from humans. In *Conference on Robot Learning (CoRL)*, 2024.

[9] Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Robotics: Science and Systems (RSS)*, 2024.

[10] Stephen Hart, Paul Dinh, and Kimberly A. Hambuchen. Affordance templates for shared robot control. In *AAAI Fall Symposium Series*, 2014.

[11] Stephen Hart, Paul Dinh, and Kimberly Hambuchen. The affordance template ROS package for robot task programming. In *IEEE International Conference on*

Robotics and Automation (ICRA), pages 6227–6234, 2015.

[12] Tairan He, Zhengyi Luo, Xialin He, Wenli Xiao, Chong Zhang, Weinan Zhang, Kris Kitani, Changliu Liu, and Guanya Shi. OmniH2O: Universal and dexterous human-to-humanoid whole-body teleoperation and learning. In *Conference on Robot Learning (CoRL)*, 2024.

[13] Tairan He, Jiawei Gao, Wenli Xiao, Yuanhang Zhang, Zi Wang, Jiashun Wang, Zhengyi Luo, Guanqi He, Nikhil Sobanbabu, Chaoyi Pan, et al. ASAP: Aligning simulation and real-world physics for learning agile humanoid whole-body skills. In *Robotics: Science and Systems (RSS)*, 2025. arXiv:2502.01143.

[14] Tairan He, Wenli Xiao, Toru Lin, Zhengyi Luo, Zhenjia Xu, Zhenyu Jiang, Jan Kautz, Changliu Liu, Guanya Shi, Xiaolong Wang, Linxi Jim Fan, and Yuke Zhu. HOVER: Versatile neural whole-body controller for humanoid robots. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 9989–9996, 2025.

[15] Industrial Digital Twin Association. Asset Administration Shell Specification – part 1: Metamodel. IDTA 01001-3-0, April 2023, 2023. URL https://industrialdigitaltwin.org/en/content-hub/aasspecifications.

[16] ISO. ISO/TS 15066:2016 robots and robotic devices—collaborative robots, 2016.

[17] ISO. ISO 10218-1:2025 robotics—safety requirements—part 1: Industrial robots, 2025.

[18] ISO. ISO 10218-2:2025 robotics—safety requirements—part 2: Industrial robot applications and robot cells, 2025.

[19] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[20] Hadas Kress-Gazit, Kerstin Eder, Guy Hoffman, Henny Admoni, Brenna Argall, Rüdiger Ehlers, Christoffer Heckman, Nils Jansen, Ross Knepper, Jan Křetínský, Shelly Levy-Tzedek, Jamy Li, Todd Murphey, Laurel Riek, and Dorsa Sadigh. Formalizing and guaranteeing human-robot interaction. *Communications of the ACM*, 64(9):78–84, 2021.

[21] Liu Liu, Wenqiang Xu, Haoyuan Fu, Sucheng Qian, Qiaojun Yu, Yang Han, and Cewu Lu. AKB-48: A real-world articulated object knowledge base. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14809–14818, 2022.

[22] Soroush Nasiriany, Sean Kirmani, Tianli Ding, Laura Smith, Yuke Zhu, Danny Driess, Dorsa Sadigh, and Ted Xiao. RT-Affordance: Affordances are versatile intermediate representations for robot manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 8249–8257, 2025.

[23] Object Management Group. DDS security specification, version 1.1, 2018.

[24] ONNX. ONNX: Open neural network exchange. https://onnx.ai, 2024.

[25] Open Robotics. SDFormat: Simulation description format. http://sdformat.org, 2024.

[26] Open Source Security Foundation. SLSA: Supply-chain levels for software artifacts. https://slsa.dev, 2023.

[27] Open X-Embodiment Collaboration. Open X-embodiment: Robotic learning datasets and RT-X models. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[28] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. ROS: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[29] Ilija Radosavovic, Bike Zhang, Baifeng Shi, Jathushan Rajasegaran, Sarthak Kamat, Trevor Darrell, Koushil Sreenath, and Jitendra Malik. Humanoid locomotion as next token prediction. *Advances in Neural Information Processing Systems*, 37:79307–79324, 2024.

[30] Justin Samuel, Nick Mathewson, Justin Cappos, and Roger Dingledine. Survivable key compromise in software update systems. In *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS)*, 2010.

[31] Lui Sha. Using simplicity to control complexity. *IEEE Software*, 18(4):20–28, 2001. doi: 10.1109/MS.2001.936213.

[32] Yiyang Shao, Xiaoyu Huang, Bike Zhang, Qiayuan Liao, Yuman Gao, Yufeng Chi, Zhongyu Li, Sophia Shao, and Koushil Sreenath. LangWBC: Language-directed humanoid whole-body control via end-to-end learning. In *Robotics: Science and Systems (RSS)*, 2025.

[33] Emanuel Todorov, Tom Erez, and Yuval Tassa. MuJoCo: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5026–5033, 2012. doi: 10.1109/IROS.2012.6386109.

[34] Santiago Torres-Arias, Hammad Afzali, Trishank Karthik Kuppusamy, Reza Curtmola, and Justin Cappos. in-toto: Providing farm-to-table guarantees for bits and bytes. In *28th USENIX Security Symposium (USENIX Security 19)*, pages 1393–1410. USENIX Association, 2019.

[35] Sirui Xu, Hung Yu Ling, Yu-Xiong Wang, and Liang-Yan Gui. InterMimic: Towards universal whole-body control for physics-based human-object interactions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12266–12277, 2025.

[36] Mingqi Yuan, Tao Yu, Wenqi Ge, Xiuyong Yao, Huijiang Wang, Jiayu Chen, Bo Li, Wei Zhang, Wenjun Zeng, Hua Chen, and Xin Jin. A survey of behavior foundation model: Next-generation whole-body control system of humanoid robots. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025. arXiv:2506.20487.

[37] Weishuai Zeng, Shunlin Lu, Kangning Yin, Xiaojie Niu, Minyue Dai, Jingbo Wang, and Jiangmiao Pang. Behav-

ior foundation model for humanoid robots. *arXiv preprint arXiv:2509.13780*, 2025.

[38] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning (CoRL)*, 2023.

Appendix A
The RSP/1.0 Interaction Protocol

This appendix specifies RSP/1.0, the communication protocol that lets an interactive object advertise a signed action skill, packaged as a Robot Skill Puzzle, which robots of differing morphology discover, authenticate, check against their own morphology and safety envelope, sandbox-load, and execute. The appendix is written as an implementable specification: the planned open-source release will contain the manifest schema, canonical encoder, verifier, selector, sandbox runner, and conformance fixtures described here.

a) Design invariant.: At every layer, **the object proposes, the robot disposes**. The manifest, constraints, assets, reference trajectories, and any object-hosted controller are advisory inputs. A robot-side safety envelope (control-barrier-function / Simplex filter, per ISO 10218 / ISO/TS 15066 [2, 31, 17, 18, 16]) is the only component that may write actuator commands. A conforming executor therefore treats object data as typed suggestions, never as authority.

A. Scope and Normative Conventions

RSP/1.0 standardizes a single-object, single-interaction session. It covers discovery, manifest verification, capability negotiation, artifact fetch, controller loading, execution ticks, safety fallback, teardown, and audit. It does not standardize scene-level task planning, perception, contact-rich multi-object composition, tactile control, fleet governance, or the certification process for a robot's safety envelope.

The protocol is intentionally small. It reuses BLE/Wi-Fi Direct/mDNS for local discovery, AAS for the product manifest, mTLS or DDS-Security for the channel, TUF/in-toto/SLSA for update and provenance metadata, ONNX/WASI for bounded controller execution, and URDF/MJCF/SDF for assets when those formats suffice. The part introduced here is the binding layer: a strict manifest schema, typed observation/action interface, deterministic skill selection rule, safety intersection rule, and auditable transcript.

Normative keywords such as MUST, MUST NOT, SHOULD, and MAY are used in their usual specification sense. Field names are lower snake case. Unless otherwise stated, numeric quantities use SI units, right-handed coordinates, meters, radians, seconds, newtons, and newton-meters. The reference frame is object-local by default. Quaternion order is always $wxyz$.

B. Layered Stack

Table V lists the eight protocol layers and the existing standard each one reuses.

TABLE V
THE RSP/1.0 LAYERS AND THE STANDARD REUSED AT EACH.

Layer	Purpose	Reuses
L0	Discovery and transport bearer	BLE/Wi-Fi Direct/mDNS [5]
L1	Signed manifest and object metadata	AAS submodel [15]
L2	Identity and authenticated channel	mTLS or DDS-Security [23]
L3	Capability, morphology, and skill-set negotiation	RSP binding layer
L4	Authenticated artifact delivery	TUF/in-toto/SLSA [30, 34, 26]
L5	Typed execution interface	ONNX [24], URDF/MJCF/SDF [28, 33, 25]
L6	Runtime monitoring and safety envelope	CBF/Simplex [2, 31]
L7	Teardown and audit	in-toto attestation [34]

C. Identifiers and Package Layout

Every advertised skill has a stable URI:

```
rsp://<manufacturer>/<object_class>/<object_id>/<skill_id>@<
    semver>
```

The tuple before the version is the identity namespace. The version follows semantic-version ordering, but anti-rollback is decided from signed TUF snapshot/timestamp metadata, not from a local "highest version seen" cache.

A distributable Robot Skill Puzzle package has the following logical layout:

```
manifest.cbor              # canonical manifest bytes
manifest.cose              # detached COSE signature set
tuf/                       # root, targets, snapshot,
    timestamp metadata
assets/<sha256>.<ext>      # URDF/MJCF/SDF meshes, inertials
    , textures
references/<sha256>.cbor   # trajectories, contacts,
    affordance frames
controllers/<sha256>.*     # ONNX, WASM, or trajectory table
provenance/<sha256>.link   # in-toto/SLSA provenance
```

The package MAY be served as files, a compressed bundle, or content-addressed objects. Verification is identical in all cases because all payloads are named by digest in the signed manifest.

D. Signed Manifest (L1)

The signed artifact is canonical CBOR with detached COSE_Sign. JSON is used in the paper only as a readable diagnostic view; a verifier MUST NOT sign or verify "JSON minus the signature field", since JSON has no unique byte form and field excision is a signature-bypass pattern. A manifest is an AAS-mapped submodel whose top-level fields are listed in Table VI.

Manufacturer authority is namespaced. Each manufacturer's intermediate certificate carries X.509 `nameConstraints` binding it to its manufacturer namespace prefix, and the robot checks the manifest URI against that prefix. Without this check, one manufacturer could sign a manifest impersonating another manufacturer's object. Multi-party authorization, when

TABLE VI
REQUIRED TOP-LEVEL FIELDS IN AN RSP/1.0 MANIFEST.

Field	Meaning
`schema_version`	Exact protocol version, e.g., `1.0.0`. Major-version mismatch is fatal.
`skill_uri`	Stable URI binding manufacturer, object, skill, and version.
`object`	Object class, object instance, physical dimensions, mass range, articulated joints, and asset references.
`skill`	Verb-like skill identifier, human-readable label, semantic affordance tags, dependencies, conflicts, and priority.
`interface`	Controller flavor, required roles, observation fields, action fields, units, frames, rate, and tensor or byte shapes.
`constraints`	Keep-out zones, workspace, force/velocity/torque bounds, contact permissions, and stop conditions.
`controller`	Optional controller descriptor: type, digest, fixed input/output shapes, WCET claim, memory budget, and sandbox profile.
`references`	Optional trajectories, grasp frames, contact schedules, terminal predicates, and recovery states, all content-addressed.
`security`	Signing certificates, namespace policy, TUF target names, provenance digests, revocation policy, and expiry.

required, is encoded as k-of-n distinct COSE signatures, with the threshold policy distributed through a TUF role.

```
{
  "schema_version": "1.0.0",
  "skill_uri": "rsp://acme/bin/b42/lift@1.0.0",
  "object": {
    "object_class": "bin",
    "object_id": "b42"
  },
  "skill": {
    "skill_id": "lift",
    "requires": ["right_gripper"]
  },
  "interface": {
    "flavor": "body_part",
    "control_rate_hz": 30
  },
  "constraints": {
    "max_force_n": 80,
    "max_torque_nm": 12
  },
  "controller": {
    "kind": "onnx",
    "sha256": "c4..."
  },
  "references": [{
    "role": "path",
    "sha256": "a7..."
  }],
  "security": {
    "signer": "acme",
    "expires": "2027-07-01"
  }
}
```

E. Handshake State Machine

The normative session state machine is:

```
idle/discover
  -> connect/auth_channel
  -> fetch_manifest/verify_sig
  -> negotiate
  -> fetch_assets/provenance
  -> load_sandbox
  -> warmup/dry_tick
  -> execute
  -> teardown
```

SAFE_FALLBACK is reachable from WARMUP and EXECUTE; ABORT is reachable from any state. Two transitions are mandatory. First, EXECUTE MUST enter SAFE_FALLBACK on an asynchronous revocation notice or channel-integrity failure. Second, resumption from SAFE_FALLBACK MUST rerun WARMUP; a controller never resumes from a stale tick. Table VII lists the normative messages and the layer each belongs to.

TABLE VII
NORMATIVE RSP/1.0 MESSAGES.

Message	Layer	Required content
`Hello`	L0	Object URI prefix, bearer, protocol versions, ephemeral endpoint.
`ManifestRequest`	L1	Robot nonce, accepted protocol versions, accepted encodings.
`ManifestReply`	L1	Manifest bytes, detached signatures, TUF target name, digest.
`CapabilityToken`	L3	Robot morphology class, roles, rates, supported flavors, budgets, sandbox profiles, safety-envelope identifier, robot nonce.
`SkillAdvert`	L3	Skill URI, ABI flavor, required roles, dependencies, conflicts, footprint, WCET, digests, constraint digest.
`SkillSetPlan`	L3	Deterministic `add`, `keep`, and `drop` sets with drop reasons.
`SkillSetCommit`	L3	Robot signature over selected tuple and nonce.
`NegotiationConfirm`	L3	Object countersignature over selected tuple, robot token, nonce, and freshness timestamp.
`ArtifactRequest`	L4	Digest, byte range, compression mode, and expected media type.
`ArtifactReply`	L4	Bytes or chunks, digest, provenance reference, decompressed size.
`LoadReport`	L4/L5	Sandbox profile, pinned opset/runtime, memory arena, fixed I/O shapes, WCET warmup result.
`DryTick`	L5	One observation/action round with zero actuation, used only to validate shapes, units, and timing.
`Start/Tick/Stop`	L5/L6	Execution start, per-cycle advisory action, and terminal reason, all mediated by the robot safety filter.
`Teardown`	L7	Final state, hashes, stop reason, safety overrides, and audit pointer.

All L3–L7 messages include `session_id`, `seq`, `prev_hash`, and `msg_hash`. This creates an append-only transcript. The transcript hash, not transport-specific state, is what later audits recompute.

F. Skill-Set Negotiation and Skill Arithmetic (L3)

RSP/1.0 makes the action-skill "addition/subtraction" in Fig. 1 an auditable selection step. The robot maintains a catalog view $\mathcal{S}_{\text{catalog}}$ from discovered verb-like skill advertisements, estimates the local scene-action set $\mathcal{S}_{\text{scene}}$ from proximity and perception, and computes

$$\mathcal{S}_{\text{active}} = (\mathcal{S}_{\text{catalog}} \cap \mathcal{S}_{\text{scene}}) \setminus \mathcal{S}_{\text{unsafe}}.$$

The set $\mathcal{S}_{\text{unsafe}}$ is computed by the robot from revocation status, morphology mismatch, unsatisfied dependencies, declared conflicts, memory/latency budget overflow, unsupported controller flavor, stale freshness proof, and an empty intersection between object constraints and the robot-owned safety envelope. It is not an object-provided label.

A compliant selector is deterministic. It sorts candidate skills by `skill_uri`; budgeted selection breaks ties by declared priority and then by digest order. Its output MUST satisfy five invariants: *authenticity* (valid signature, namespace, TUF metadata, and provenance), *minimality* (only active skills are loaded), *closure* (all active dependencies are active), *consistency* (no active conflict pair), and *budget/safety* (resource budgets fit and the safe-and-reachable intersection is nonempty). The `SkillSetPlan` gives a machine-readable reason for every dropped skill. The v1.0 reasons are:

```
revoked, role_missing, dependency_missing, conflict, budget,
safety_empty, abi_unsupported, stale, policy_denied
```

G. Typed Execution Interface (L5)

The interface I from Section II is typed at the byte and tensor level. It declares the controller flavor, observation fields, action fields, frame for every pose/vector quantity, unit for every scalar quantity, control rate, alignment, endianness, fixed tensor shapes, and allowed missing-data policy. A robot MUST reject a manifest if any field required by the controller cannot be produced by its estimator or morphology mapper. Table VIII lists the three execution flavors.

TABLE VIII
RSP/1.0 EXECUTION FLAVORS. NONE MAY WRITE ACTUATOR COMMANDS DIRECTLY.

Flavor	Semantics
TRAJECTORY	Produces a time-indexed reference path or contact schedule for a robot-owned tracker.
BODY_PART	Produces role-keyed task-space targets, e.g., hand, foot, pelvis, or base targets.
SKILL	Produces a closed-loop task-space action with declared observation/action fields and rate.

The primitive field types are `bool`, signed/unsigned integers, `float32`, `float64`, `vec3`, `quat_wxyz`, `pose_se3`, `twist6`, `wrench6`, `joint_vector`, `image`, and fixed-shape tensors. Delta-pose actions are velocities (Δ per $1/$`control_rate_hz`) with zero-order hold. This removes the ambiguity of whether a delta accumulates over controller ticks or wall-clock time under rate mismatch.

Observation fields are tagged as robot proprioception, robot estimate, object state estimate, or object reported. A robot MUST treat object-reported values as untrusted measurements unless it has an authenticated sensor path and an estimator that explicitly accepts them. Multi-end-effector roles carry a relative-frame block stating whether the chain is rigidly closed. Base-aligned roles carry a required workspace volume. The v1.0 guarantee is scoped to single- and dual-end-effector reach-and-track; richer tactile/contact interfaces are future work.

H. Artifact Delivery and Sandbox Loading (L4)

Artifacts are fetched only after a successful `SkillSetCommit`. Each artifact is content-addressed by SHA-256, has a declared decompressed size, and is covered by TUF target metadata. A robot MUST verify digest, size, media type, TUF freshness, provenance link, and sandbox profile before loading the bytes. External-data references, dynamic library loading, network access, filesystem write access, and undeclared host calls are disabled by default.

Object-supplied controllers run in a WASI sandbox with a closed host-function set. For ONNX controllers, the runtime is inside the sandbox; custom operators are disabled, opset is pinned, tensor arenas are capped, and all shapes are fixed. Fixed shapes are necessary for bounded execution but are not treated as the security boundary. The sandbox emits advisory task-space outputs only; the real-time safety thread consumes those outputs through the L6 filter.

I. Security and Freshness

The channel is mutually authenticated (L2). Downgrade protection is not delegated to the TLS transcript hash: the object returns a signed confirmation over

$$H(\mathit{CapabilityToken} \parallel \mathit{SkillSetPlan} \parallel \mathit{SkillSetCommit} \parallel \mathit{nonce} \parallel \mathit{timestamp}),$$

and the robot verifies that the selected flavor is the strongest mutually supported flavor under its local policy. This binding is transport-independent, so it holds for both mTLS and DDS-Security.

Freshness uses a robot-generated challenge nonce. A nonce embedded in a cached manifest is a constant and proves only that the manifest was once signed. Revocation anchors (OCSP, CRL, and TUF endpoints) are taken from the trust root, never from the artifact being validated. Long-running sessions re-check revocation on a fixed cadence and immediately enter SAFE_FALLBACK on a negative result. This exposes an unavoidable deployment tension: offline caching and live proof of key possession cannot both be guaranteed when the object is fully offline; a robot policy must choose the acceptable mode for its operating environment.

J. Safety Integration (L6)

The object's `keep_out_zones`, force limits, velocity limits, contact permissions, and workspace bounds are advisory inputs to the robot-owned safety envelope. The executed admissible set is the intersection of the robot's certified safe set with the retargeted object constraints. If the intersection is empty, the skill is dropped before loading. If it becomes empty during execution, the robot enters SAFE_FALLBACK.

Object-hosted code is never scheduled on the hard-real-time actuator thread. It runs off the real-time path at an advisory rate; the real-time thread tracks the last valid task-space action through the CBF/Simplex filter. A controller that misses its declared WCET budget during WARMUP is rejected. During execution, a missed advisory tick triggers hold-last semantics in force space, not position space, so a stall during contact does not command an increasing force. The tolerated number of missed ticks is derived from the safety envelope's margin, not from a fixed round number.

K. Error Model

Errors are structured records:

```
{
  "layer": "L3",
  "code": "E_ROLE_MISSING",
  "recoverable": false,
  "detail": "required role right_gripper is unavailable",
  "transcript_hash": "3a..."
}
```

A robot MUST fail closed on every unrecognized error code. The standard codes are:

```
E_VERSION_UNSUPPORTED, E_SIG_INVALID, E_NAMESPACE,
     E_ROLLBACK,
E_REVOKED, E_FRESHNESS, E_ABI_UNSUPPORTED, E_ROLE_MISSING,
E_DEPENDENCY, E_CONFLICT, E_BUDGET, E_SAFETY_EMPTY,
E_ARTIFACT_DIGEST, E_PROVENANCE, E_SANDBOX, E_SHAPE,
E_WCET, E_TIMEOUT, E_REVOCATION_ASYNC, E_POLICY_DENIED
```

L. Versioning and Compatibility

Major versions are not wire-compatible. A robot MUST reject a manifest with an unsupported major version. Minor versions are backward compatible only when every unknown field is explicitly marked `optional`; unknown required fields are fatal. Patch versions may fix metadata or conformance wording but MUST NOT change wire semantics. Extension fields are namespaced as `x_<vendor>_<field>` and are ignored unless a robot policy explicitly enables that vendor extension.

M. Audit Record (L7)

At teardown the robot writes an audit record containing the selected skill URI, manifest digest, artifact digests, trust-root version, capability token hash, negotiation transcript hash, sandbox profile, safety-envelope identifier, execution duration, stop reason, safety overrides, missed advisory ticks, and revocation status. The audit record does not include raw private sensor streams by default. A manufacturer or regulator can replay verification from the transcript and trust roots without rerunning the controller.

N. Conformance Suite

An implementation is considered conforming only if it passes the following fixture groups:

1) Golden manifest decoding and signature verification, including negative cases for invalid canonical encoding, bad signatures, namespace mismatch, threshold failure, expiry, and TUF rollback.
2) Deterministic L3 selection fixtures for add/keep/drop, dependency closure, conflict rejection, revoked-skill subtraction, budget overflow, unsupported ABI, stale freshness proof, and empty safety intersection.
3) Artifact-delivery fixtures for digest mismatch, decompression bomb, undeclared external data, malformed ONNX graphs, custom-operator rejection, sandbox escape attempts, and oversized tensor arenas.
4) Interface fixtures for frame conversion, quaternion order, SI-unit conversion, delta-pose velocity semantics, tensor shape checking, missing observation fields, and role-to-morphology mapping.
5) Runtime fixtures for WCET violation, jitter, missed ticks, revocation during execution, safe fallback, hold-last force behavior, and CBF override audit confirming that actuator authority never leaves the robot filter.
6) Reproducibility fixtures that recompute the L3 plan, commit, confirmation, and final audit hash from the transcript alone.

O. Open Problems

Several structural issues are not solved by RSP/1.0. First, the safety claim ultimately depends on a forward-invariance guarantee for the robot-owned envelope under arbitrary adversarial advisory input. Second, real-time guarantees for untrusted code running alongside a control loop are architectural, not just benchmarking details; this is why the controller is kept off the hard-real-time thread. Third, offline caching and live freshness are in tension. Fourth, floating-point nondeterminism across WASM/ONNX runtimes makes identical cross-platform controller outputs unrealistic without a stricter numerical profile. Fifth, federated manufacturer PKI and namespace governance are social and operational problems, not only cryptographic ones. Sixth, reputation-driven revocation can itself be attacked by spoofed telemetry and needs authenticated, rate-limited reporting. Finally, v1.0 is normatively single-interaction: composition and contention among multiple simultaneously present objects require additional scheduling, coupling, and safety semantics.