SELECTIVE EXECUTION WITH ANSWER-DETERMINED TERMINATION FOR STRUCTURED REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reasoning systems spend more computation on problems that look harder (Snell et al., 2025; Wu et al., 2025; Muennighoff et al., 2025), yet how much a problem needs is fixed by the answer it asks for. Once every explanation compatible with the observations implies the same answer, further reasoning cannot change it. Solvers routinely continue past this point, recovering causal structure (Zhang et al., 2021), proofs (Trinh et al., 2024), or complete candidate scores (Benny et al., 2021) that the question never required; we call this *over-solving*. Here we formalize answer determination under incomplete information, in the sense of certain answers (Imieliński & Lipski, 1984) and partial identification (Zhang, 2024), and show that an answer can be unique while arbitrarily many richer explanations remain open, so a solver can stop earlier and can answer instances whose explanation is unrecoverable. Because possible answers can contract in any monotone pattern, no task-independent rule says which computation to skip; we therefore terminate execution with a task-specific completion rule that fires once the answer is determined, and order the computations before that point with a learned answer-relevance energy. On ACRE (Zhang et al., 2021), CLUTRR (Sinha et al., 2019), AlphaGeometry (Trinh et al., 2024), RAVEN-FAIR (Benny et al., 2021), and MNS (Zhang et al., 2020), this selective execution omits 33.95% to 99.98% of task-specific computation while matching or improving baseline accuracy, with gains of up to 30.99 percentage points. Efficient reasoning depends not only on how fast computations run but on which computations the answer requires.

1 INTRODUCTION

A solver for a structured reasoning task, one whose semantics specify how rules, relations, or latent variables determine the answer, usually answers a query by recovering something larger than the answer. To decide whether a set of objects activates a machine, a causal-induction solver fits a complete causal model (Zhang et al., 2021); to prove one geometric target, a deduction engine expands the proof state until the target appears (Trinh et al., 2024); to pick one candidate panel, a visual-reasoning model scores every candidate in full (Benny et al., 2021). The query rarely requires the whole object. Once every explanation still compatible with the observations implies the same answer, further computation cannot change it, and everything spent after that point is waste. We call this *over-solving*.

Adaptive computation has become a central concern in reasoning with language models, where more inference-time work improves accuracy (Wei et al., 2022; Wang et al., 2023; Yao et al., 2023; Snell et al., 2025; Muennighoff et al., 2025) and easy problems provoke far longer traces than they need (Chen et al., 2024; Sui et al., 2025). The stopping rules developed there read signals from the model, such as agreement among sampled answers (Aggarwal et al., 2023), convergence of an intermediate answer (Liu & Wang, 2025), estimated difficulty (Snell et al., 2025; Yu et al., 2025), or a learned choice between short and long reasoning (Fang et al., 2025); rational metareasoning values a computation by its expected effect on the decision (Russell & Wefald, 1991; Hay et al., 2012). Over-solving is defined by the problem instead. The question is not whether the solver's outputs have stabilized but whether the problem still permits more than one answer.

Fig. 1 makes the distinction concrete in the blicket-machine setting of ACRE (Zhang et al., 2021). The machine turns on whenever at least one activating object is present. The observations establish

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

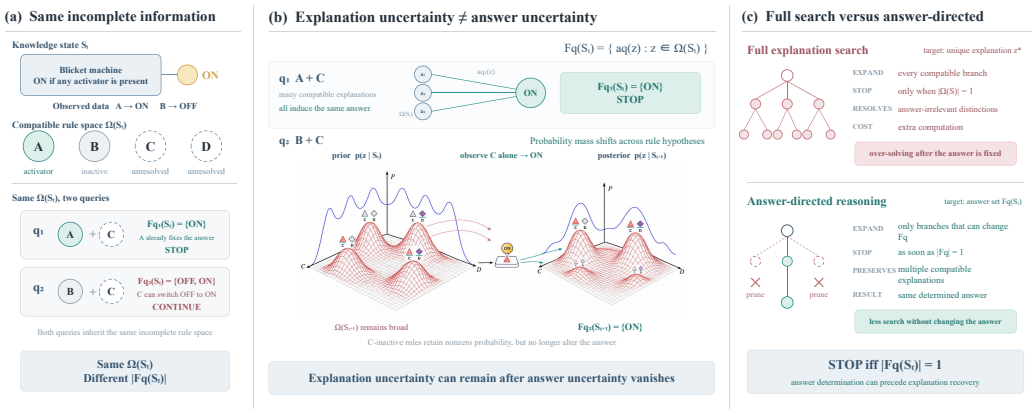

Figure 1: **Answer relevance of unresolved information.** Under the same observations, the role of C is unresolved for both queries. It cannot change the answer for $A + C$, because A alone activates the machine, but it decides the answer for $B + C$.

that A activates the machine and that B does not, and leave the roles of C and D open. For the query $A + C$, the role of C is irrelevant, because A alone turns the machine on; for $B + C$, the same open role decides the answer. What remains uncertain about the explanation need not remain uncertain about the answer, and which uncertainty matters depends on the query.

We formalize this through the explanations that remain compatible with the information acquired so far and the answers they induce, a construction related to partial identification (Zhang, 2024; Joshi et al., 2024) and to certain answers over incomplete information (Imieliński & Lipski, 1984). An answer is determined when all compatible explanations agree on it. We show that the answer can be determined while arbitrarily many richer results, whether rule systems, programs, causal structures, or proofs, remain possible. Stopping at answer determination therefore lets a solver reason less, and it lets the solver solve more, because some instances have a unique answer under information that identifies no unique explanation. We also show that the set of possible answers contracts monotonically as information accumulates and that every monotone contraction ending in a single answer can occur. No task-independent law says which computation eliminates an answer or when the last alternative disappears; that knowledge lives in the task structure.

These results shape the method. Selective execution requires two decisions, when to stop and what to compute next, and the two have different characters. A task-specific completion rule, derived from the task semantics and sound by construction, decides when to stop, terminating execution as soon as the acquired information fixes the answer, whether or not the richer result has been recovered. A learned, state-conditioned answer-relevance energy decides what to compute next, ordering the candidate computations so that the completion rule fires with less work, and it is re-evaluated as execution changes which computations remain useful. We instantiate the framework on ACRE (Zhang et al., 2021), CLUTRR (Sinha et al., 2019), AlphaGeometry (Trinh et al., 2024), RAVEN-FAIR (Benny et al., 2021), and MNS (Zhang et al., 2020). The omitted computation differs in each case. Set inclusion replaces causal-model fitting, relation composition avoids proof search, the target narrows geometric rule matching, score bounds prune candidate comparisons, and agreement across rule-wise optima makes weight optimization unnecessary.

Across the five tasks, selective execution omits 33.95% to 99.98% of task-specific computation and yields measured speedups of 2.34× to 72,763× over the original solvers, while accuracy is maintained or improved by up to 30.99 percentage points. Ablations separate the two components. Termination alone accounts for most of the savings where the task structure leaves little ordering freedom, and the learned ordering raises the savings from 30.34% to 54.63% where it does not.

2 Related Work

Scaling reasoning at inference time. Language models reason better when allowed more computation. Chain-of-thought prompting exposes intermediate steps (Wei et al., 2022), self-consistency aggregates several sampled chains (Wang et al., 2023), Tree of Thoughts searches over partial so-

lutions (Yao et al., 2023), and STaR bootstraps training from successful rationales (Zelikman et al., 2022). Process supervision scores individual steps (Lightman et al., 2024), reinforcement learning induces reflection and verification without demonstrations of those behaviors (Guo et al., 2025), and scaling studies compare inference strategies under matched budgets (Snell et al., 2025; Wu et al., 2025) or set the budget directly (Muennighoff et al., 2025). The same literature documents overthinking, in which models produce far longer traces than easy problems need (Chen et al., 2024; Sui et al., 2025), and diagnoses the excess from the trace. Over-solving is defined by the task instead, as computation performed after every explanation compatible with the available information already agrees on the answer.

Adaptive computation and stopping. Rational metareasoning ranks computations by their expected effect on the final decision (Russell & Wefald, 1991; Hay et al., 2012). Recent stopping rules read the model instead. Adaptive-Consistency stops sampling when answers agree (Aggarwal et al., 2023), compute-optimal scaling predicts which instances benefit from a larger budget (Snell et al., 2025), and Answer Convergence tests whether the answer survives truncation of the trace (Liu & Wang, 2025). Allocation can also be learned. Thinkless chooses between short and long reasoning (Fang et al., 2025), inference-aware optimization trains models to respect a budget (Yu et al., 2025), and length-regularized objectives reward shorter correct responses (Arora & Zanette, 2025). A separate line lowers the cost of each reasoning step through continuous-space compression and speculative reasoning (Shen et al., 2025; Pan et al., 2025). Answer determination is defined by the task constraints rather than by model behavior. Further computation is unnecessary once every compatible explanation agrees on the answer, whether or not the model's outputs have stabilized.

Reasoning with incomplete information. Active feature acquisition chooses which unobserved features to obtain for a prediction, trading their value against their cost (Li & Oliva, 2021; Norcliffe et al., 2025), and language-based variants let a model request missing evidence or ask questions that reduce uncertainty (Zhou et al., 2025; Wang et al., 2025). Partial identification retains the set of causal quantities or policies consistent with the available assumptions and data (Zhang, 2024; Joshi et al., 2024), and a query over an incomplete database has a certain answer when every completion agrees on it (Imieliński & Lipski, 1984). Answer determination is the same criterion applied to a reasoning process. We seek the least-cost computation that reduces the set of possible answers to a singleton, and we treat the steps of a solver, rather than features or database tuples, as the information being acquired.

Structure in symbolic reasoning. Symbolic and neuro-symbolic solvers use task structure to control search. Conditional theorem provers (CTPs) learn which rules to apply for a goal rather than exploring every proof path (Minervini et al., 2020), program synthesis exploits compositional rules and reusable libraries (Nye et al., 2020; Ellis et al., 2021), and AlphaGeometry pairs learned construction proposals with symbolic deduction (Trinh et al., 2024). These systems organize search around recovering a proof or a program. Our objective is narrower. We retain only the computation needed to fix the requested answer, and we stop even when the proof, program, or rule system remains unresolved.

3 Theory of Over-Solving

We first define when the acquired information determines the answer, then separate this requirement from resolving a richer result, and finally characterize how the set of possible answers can evolve as reasoning proceeds.

3.1 When Is the Answer Determined?

Consider a fixed reasoning problem. Let Ω be the set of complete explanations consistent with the task specification and $z^\star \in \Omega$ the true one. A complete explanation fixes the rules, relations, or latent variables of the problem and thereby fixes the answer. Let I index the information items that characterize explanations, so that each $z \in \Omega$ is written $z = (z_i)_{i \in I}$ and resolving every item identifies $z^\star$.

Reasoning resolves these items progressively. When the items in $S \subseteq I$ have been resolved, the explanations compatible with the acquired values $z_S^\star$ form

$$\Omega_S = \{z \in \Omega : z_S = z_S^\star\}.$$

Additional information removes incompatible explanations and never removes the true one. Let $a : \Omega \to \mathcal{Y}$ be the answer map into the answer space $\mathcal{Y}$. The answers that remain possible under the current information are

$$F(S) = \{a(z) : z \in \Omega_S\}.$$

Definition 3.1 (Answer determination)**.** The answer is determined under information S if $|F(S)| = 1$.

The condition asks every compatible explanation to agree on the requested answer; it does not ask the explanation itself to be unique. When $|F(S)| > 1$, the available information still admits different answers. Once $|F(S)| = 1$, no further information can change the answer. In Fig. 1, the role of C admits several explanations for $A + C$, all of which imply that the machine turns on. Answer determination is therefore a completion criterion for the current question rather than for the problem as a whole.

Enumerating $F(S)$ directly can cost as much as the reasoning it is meant to cut short. We therefore check answer determination through a task-specific completion rule $\mathrm{Done}_b(x, S)$, where b indexes the task and x the instance. The rule is sound if

$$\mathrm{Done}_b(x, S) \implies |F_x(S)| = 1,$$

in which case it certifies that the answer is determined and permits termination while the explanation is still unresolved. Sec. B gives the rule for each task and states where a rule is operational rather than exact.

3.2 Answer Determination Precedes Richer Resolution

A solver may pursue a result richer than the answer, such as a rule system, a program, a causal structure, or a proof. Let $r : \Omega \to \mathcal{R}$ represent this result and assume a map $q : \mathcal{R} \to \mathcal{Y}$ with $a = q \circ r$, so that the richer result suffices to obtain the answer. Under information S, the richer results that remain possible are

$$R(S) = \{r(z) : z \in \Omega_S\}, \qquad F(S) = q(R(S)).$$

Different richer results can map to the same answer through q, so distinguishing among them may exceed what the current question requires.

Proposition 3.2 (Separation between answers and richer results)**.** *For any information state S, $|R(S)| = 1$ implies $|F(S)| = 1$, but not conversely. Moreover, for every integer $K \geqslant 1$ there exist a reasoning problem and an information state S with $|R(S)| = K$ and $|F(S)| = 1$.*

Proof. The implication follows from $F(S) = q(R(S))$. For the second claim, take $I = \{1\}$, let Ω consist of K explanations with distinct information values, let r be the identity, and let q be constant. At $S = \varnothing$ all K richer results remain possible while the answer is already unique. □

Determining the richer result is thus sufficient for determining the answer, but arbitrarily many richer results can remain undistinguished after the answer is fixed, and continuing to distinguish among them is over-solving. The separation has two consequences, one for how long an individual problem must be reasoned about and one for which problems can be answered at all. For a reasoning process that resolves information progressively, $S_0 \subseteq \cdots \subseteq S_T$, define the completion times

$$\tau_{\mathrm{ans}} = \inf\{t : |F(S_t)| = 1\}, \qquad \tau_{\mathrm{rich}} = \inf\{t : |R(S_t)| = 1\},$$

with the infimum of the empty set taken as $+\infty$. For a collection of instances x, each with available information S_x, define

$$\mathcal{Q}_{\mathrm{rich}} = \{x : |R_x(S_x)| = 1\}, \qquad \mathcal{Q}_{\mathrm{ans}} = \{x : |F_x(S_x)| = 1\}.$$

Corollary 3.3 (Reason less, solve more)**.** *$\tau_{\mathrm{ans}} \leqslant \tau_{\mathrm{rich}}$ and $\mathcal{Q}_{\mathrm{rich}} \subseteq \mathcal{Q}_{\mathrm{ans}}$, and both relations can be strict.*

Proof. Both follow from Proposition 3.2. In its construction with $K = 2$, $S_0 = \varnothing$, and $S_1 = I$, we have $\tau_{\text{ans}} = 0 < 1 = \tau_{\text{rich}}$, and under S_0 the instance lies in $\mathcal{Q}_{\text{ans}}$ but not in $\mathcal{Q}_{\text{rich}}$. □

The first relation says that the requested answer can be obtained while the richer result remains unresolved, so reasoning can finish earlier and the solver can *reason less*. The second says that, under the same information, some problems already have a unique answer although their richer result cannot be recovered, so the solver can *solve more*.

3.3 How Possible Answers Evolve

Determining the answer can thus require less work than determining the richer result. To identify which work can be omitted, we ask how the possible answers change as information increases and whether their evolution obeys laws that could guide computation selection.

Proposition 3.4 (Monotonicity of possible answers)**.** *For any $S \subseteq S' \subseteq I$, $F(S') \subseteq F(S)$.*

Proof. Additional information can only eliminate compatible explanations, so $\Omega_{S'} \subseteq \Omega_S$, and applying a to both sets gives the claim. □

Monotonicity means that new information cannot reintroduce an eliminated answer, but it does not say when the answer set shrinks or by how much. The following theorem shows that nothing else is guaranteed. Subject to the terminal condition that complete information determines the answer, every monotone evolution can occur.

Theorem 3.5 (Characterization of possible-answer evolution)**.** *Let I be a finite index set and $\mathcal{Y}$ an answer space. A mapping $G : 2^I \to 2^{\mathcal{Y}}$ is realized as $F(S) = G(S)$ for all $S \subseteq I$ by some reasoning problem in the setup above if and only if $|G(I)| = 1$ and $G(S') \subseteq G(S)$ whenever $S \subseteq S'$.*

Proof sketch. Necessity follows from Proposition 3.4 and $\Omega_I = \{z^\star\}$. For sufficiency, let $G(I) = \{y^\star\}$ and let $z^\star$ take a common value at every item. For each $A \subsetneq I$ and each $y \in G(A)\backslash\{y^\star\}$, add an explanation that agrees with $z^\star$ exactly on A and has answer y. Such an explanation is compatible with S exactly when $S \subseteq A$, so $F(S)$ collects $y^\star$ together with every $G(A)\backslash\{y^\star\}$ for $A \supseteq S$, which equals $G(S)$ by monotonicity. Sec. E gives the full construction. □

Possible answers may therefore vanish one at a time, or persist through many steps until a single information item collapses the set. Determination and monotonicity impose no further constraint on how much each computation should eliminate, so which computation is worth performing must come from the task, from how its rules, relations, and intermediate quantities jointly constrain the requested answer. The next section takes answer determination as the completion requirement and learns to select computation under it.

4 Selective Execution with Answer-Determined Termination

Sec. 3 shows that an answer can be determined while richer explanations remain open, and that no task-independent law says which computation closes the gap. Selective execution therefore rests on two components with different roles. A task-specific completion rule decides when to stop and, when sound, guarantees that early termination returns the same answer as complete reasoning. A learned answer-relevance energy decides what to compute next and orders the candidate computations so that the completion rule is reached with less work. Fig. 2 gives an overview.

4.1 From Answer Determination to Computation Preference

For an instance x, query included, let S be the information acquired so far and $\mathcal{C}_x(S)$ the candidate computations currently available. A computation may be a local evaluation, a rule application, or a group of dependent operations. Let $\mathfrak{M}_x$ be the set of executable plans, with $S_x(M)$ the information that a plan M acquires and $C_x(M)$ its cost. We seek a least-cost plan that determines the answer,

$$M_x^\star \in \arg\min_{M \in \mathfrak{M}_x} C_x(M) \quad \text{s.t.} \quad |F_x(S_x(M))| = 1.$$

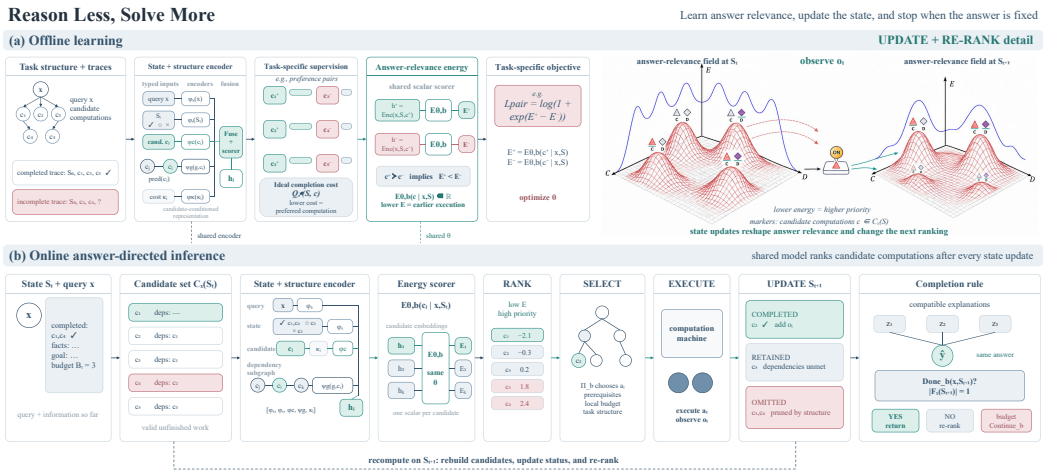

Figure 2: **Selective execution with answer-determined termination.** A state-conditioned answer-relevance energy orders the candidate computations, execution updates the reasoning state, and the ordering is recomputed as new information arrives. A task-specific completion rule terminates execution once the answer is determined, leaving computation that cannot affect the answer unexecuted.

To turn plan selection into a choice of the next computation, consider an unresolved state S. Let $\mathfrak{M}_x(S; c)$ contain the continuation plans whose first computation is c, and define

$$Q^\star_x(S, c) = \inf\{C_x(M \mid S) : M \in \mathfrak{M}_x(S; c),\ |F_x(S_x(S, M))| = 1\},$$

where $S_x(S, M)$ is the state reached by executing M from S and $C_x(M \mid S)$ counts the cost of c and of all subsequent computations; the infimum is $+\infty$ when no such plan exists. This ideal completion cost induces the preference $c_i \succ^\star_{x,S} c_j$ whenever $Q^\star_x(S, c_i) < Q^\star_x(S, c_j)$. A computation is preferred when starting with it permits lower-cost determination, which credits both computations that settle the query directly and those that establish useful prerequisites. Because $Q^\star_x$ depends on the true explanation, it is unavailable at inference time; we take it as the learning target and construct supervision from training trajectories and task structure.

4.2 Learning an Answer-Relevance Energy

We parameterize computation priority with an energy $E_{\theta,b}(c \mid x, S) \in \mathbb{R}$, where b indexes the task and θ the parameters. For a fixed instance and state, the energy defines an answer-relevance field over the candidate computations, with lower energy marking a computation to execute earlier; the field approximates the ordering induced by $Q^\star_x$.

Supervision comes from the task itself. Training trajectories, structural annotations, and candidate evaluations expose dependency, sufficiency, and compatibility relations between computations and the query, and these relations yield, depending on the task, a relevance label for a candidate, the identity of the next computation to prioritize, or a relative order between two computations. A computation that supplies a prerequisite of the target, for instance, ranks above one that develops an unrelated branch, which rewards useful computations before their effect appears in the possible-answer set.

Let $\mathcal{D}_b$ be the resulting supervision for task b, each example pairing a state with a signal ξ. The energy is trained by

$$\min_\theta\ \mathcal{L}_b(\theta) = \mathbb{E}_{(x,S,\xi)\sim\mathcal{D}_b}\left[\ell_b\big(\{E_{\theta,b}(c \mid x, S)\}_{c\in\mathcal{C}_x(S)}, \xi\big)\right], \tag{1}$$

where ℓ_b is the loss matching the form of ξ, such as a logistic loss for relevance labels or a pairwise ranking loss for orderings. Secs. B and C give the supervision and objective for each task. At inference time the energy is evaluated on the task input and the information acquired so far, so the ordering adapts as new results change which computations remain useful.

Algorithm 1 Selective execution with answer-determined termination

Require: Instance x, energy $E_{\theta,b}$, planner Π_b, completion rule Done_b, continuation procedure $\mathrm{Continue}_b$, budget B

Training: construct supervision $\mathcal{D}_b$ from training data and task structure; train $E_{\theta,b}$ with Eq. (1)
Inference: initialize S from x, including computation status
while $\neg\,\mathrm{Done}_b(x, S)$ **do**
  $\mathcal{C} \leftarrow \mathcal{C}_x(S)$
  if B is exhausted **or** $\mathcal{C} = \varnothing$ **then**
    return $\mathrm{Continue}_b(x, S)$
  end if
  $R \leftarrow \mathrm{Rank}_{E_{\theta,b}}(\mathcal{C} \mid x, S)$
  $a \leftarrow \Pi_b(x, S, R)$
  $o \leftarrow \mathrm{Execute}(a)$
  $S \leftarrow T_x(S, a, o)$; record structurally justified omissions in S
  update B
end while
return $\mathrm{Answer}_b(x, S)$

4.3 Energy-Guided Selective Execution

The executor pursues answer determination from the current state. A planner combines the learned ordering with the task's dependencies to choose the next action,

$$a_t = \Pi_b\big(x, S_t, \mathrm{Rank}_{E_{\theta,b}}(\mathcal{C}_x(S_t))\big),$$

where Rank orders the candidates from low to high energy with a fixed tie-breaking rule. An action executes one computation or a group that shares prerequisites or intermediate results; selecting the lowest-energy candidate is the sequential case. Executing a_t yields an observation o_t and updates the state, $S_{t+1} = T_x(S_t, a_t, o_t)$. New information may enable computations or establish that others can no longer affect the answer. Eliminating a candidate removes its remaining evaluations, and establishing the target makes other derivations unnecessary. The executor keeps unselected candidates for later, records completed and omitted computations in S so that later candidate sets reflect the work already resolved, and re-evaluates the energy on the updated state.

The completion rule $\mathrm{Done}_b(x, S)$ of Sec. 3 is checked at initialization and after every update, so execution ends as soon as the answer is determined even when richer explanations remain open. Where the rule is sound, no further computation could have changed the returned answer; where it is operational, termination is a decision of the executor rather than a certificate, and Sec. B says so for each task. Scoring models fit the same construction once unevaluated score terms are treated as unresolved information.

A budget B bounds the selective stage. When it is exhausted or no candidate remains, the continuation procedure $\mathrm{Continue}_b(x, S)$ completes the remaining computation or invokes the task's original procedure and returns its result or an unresolved status. Alg. 1 summarizes training and inference.

4.4 Instantiation on Five Tasks

Tab. 1 summarizes the five instantiations. In each, the baseline recovers a richer result than the query requires, the completion rule detects when the answer is fixed before that result is recovered, and the energy orders the computations that tend to reach the rule. The instantiations differ only in these three task-specific pieces; the executor of Alg. 1 is shared.

5 Experiments

Setup. We evaluate on ACRE (Zhang et al., 2021), CLUTRR (Sinha et al., 2019), AlphaGeometry (Trinh et al., 2024), RAVEN-FAIR (Benny et al., 2021), and MNS (Zhang et al., 2020), against neuro-symbolic optimization (NS-Opt) (Zhang et al., 2021), CTP (Minervini et al., 2020), deductive database with algebraic reasoning (DDAR) (Trinh et al., 2024), multi-scale relation network (MRNet) (Benny et al., 2021), and the original MNS solver (Zhang et al., 2020), respectively. Sec. A describes the benchmarks and Sec. C the implementation, the timing protocol, and the baseline runs. We report three quantities. Accuracy is task accuracy on the evaluation split. Work

Table 1: **Instantiation on five tasks.** Each row gives the computation that the energy orders, the completion rule, and the continuation procedure for one task and its baseline. Sec. B gives the full constructions.

Task / baseline	Energy-ordered unit	Completion rule	Continuation
ACRE *NS-Opt*	Observation-query check	$T \subseteq Z$, or $R_i \subseteq T$ for a processed observation i	Remaining observations; otherwise *undetermined*
CLUTRR *CTP*	One relation-composition step along a path	The current compositions determine a unique relation between the queried entities	CTP proof search
AlphaGeometry *DDAR*	One rule application or algebraic check	The target proposition is derived	Unrestricted DDAR
RAVEN-FAIR *MRNet*	One local-unit evaluation for one candidate	A fully scored candidate exceeds the upper bounds of all remaining candidates	Remaining unit evaluations
MNS *Original solver*	One rule-expression pair verification	The accepted rules yield a unique winner for the query	Original MNS solver on the current query

Table 2: **Accuracy and computational gains on five reasoning tasks.** Accuracy compares each instantiated executor with its baseline on the same evaluation split. Work reduction is the fraction of task-specific unit computations that selective execution omits. System speedup is the wall-clock ratio of the baseline to our executor for the solving stage and includes gains from reformulating the baseline's computation as well as from selective execution.

Task	Baseline	Baseline Acc.	Ours Acc.	Δ Acc.	Work Reduction	System Speedup
ACRE	NS-Opt	69.01%	**100.00%**	+30.99 pp	41.66%	72,763×
CLUTRR	CTP	76.24%	**95.32%**	+19.08 pp	72.80%	972.23×
AlphaGeometry	DDAR	88.24%	88.24%	0.00 pp	33.95%	2.34×
RAVEN-FAIR	MRNet	87.36%	**99.59%**	+12.23 pp	54.63%	14.41×
MNS	Original solver	75.00%	75.00%	0.00 pp	99.98%	84.79×

reduction is the fraction of task-specific unit computations, named for each task in Sec. B, that selective execution omits relative to full execution. System speedup is the ratio of the baseline's wall-clock time to ours for the solving stage on the same hardware; because an instantiation may also replace the baseline's computation with a cheaper kind, this ratio reflects the reformulation as well as selective execution, and work reduction is the direct measure of the latter.

Main results. Tab. 2 summarizes the results. On all five tasks the instantiated executor matches or improves the accuracy of its baseline while omitting between a third and nearly all of the task-specific computation. The two computational columns answer different questions. Work reduction counts the unit computations that selective execution omits within the instantiated procedure, whereas system speedup compares wall-clock time against a baseline that may perform a different and costlier kind of computation. On ACRE, for instance, most of the speedup comes from replacing continuous causal-model optimization with set comparisons, and selective execution adds the 41.66% of observation checks that the completion rule makes unnecessary.

Where the gains come from. The five tasks over-solve in different ways, and the omitted computation differs accordingly.

ACRE. NS-Opt infers a latent causal model from the observed object combinations and then reads the queried state off that model, resolving causal roles that many queries do not need. Under the OR mechanism, two set-inclusion conditions on the processed observations fix the machine state without resolving the remaining roles (Sec. B.1). The energy orders the observations by how likely each is to satisfy a condition for the current query, and once one holds, the remaining checks and the causal inference of the original procedure are omitted. The rule is exact, and it is complete once all observations are processed, so the accuracy gain comes from queries on which causal fitting fails although the answer is determined. The largest gap is on *undetermined* queries, whose label itself states that $|F(S)| > 1$ after all evidence (Sec. D).

Table 3: **Ablation of termination and computation ordering.** The learned, random, and fixed heuristic orderings share the completion rule and differ only in the order of computation; full evaluation disables termination. Work reduction and speedup are measured against full evaluation of the instantiated executor rather than against the baseline, so the AlphaGeometry values differ from Tab. 2, and speedup includes the cost of computing the ordering.

Variant	AlphaGeometry			RAVEN-FAIR		
	Acc.	Work Red.	Speedup	Acc.	Work Red.	Speedup
Learned energy (ours)	88.24%	36.69%	2.255×	99.59%	54.63%	1.212×
Random ordering	88.24%	35.64%	2.247×	99.59%	30.34%	1.131×
Fixed heuristic ordering	88.24%	35.97%	2.273×	99.59%	52.37%	1.236×
Full evaluation	88.24%	0.00%	1.000×	99.59%	0.00%	1.000×

CLUTRR. CTP runs a neural proof search separately for each candidate relation over the relation graph. Composition rules learned from training proofs, applied along a path between the queried entities, yield the queried relation directly; the energy orders the executable composition steps, and execution stops when the compositions determine a unique relation (Sec. B.2). Only queries that the compositions leave open are passed to CTP. The savings grow with chain length and reach about 75% on the longest chains (Sec. D).

AlphaGeometry. DDAR expands the proof state by matching every applicable rule, enumerating bindings, and deriving new relations until the target appears, which produces many valid conclusions that the target does not need. Training proofs expose which rules tend to advance a given target from given constructions; the energy orders rule applications accordingly, and existing algebraic relations may establish the target directly (Sec. B.3). Once the target is derived, further matching, binding enumeration, and dependency processing are omitted. The executor solves exactly the problems that DDAR solves, 45 of 51, with a third fewer predicate checks.

RAVEN-FAIR. MRNet scores each candidate with an integrated multiscale relational model. We decompose the score into local units by component and attribute, learn their rule responses, and evaluate the units in energy order (Sec. B.4). Because every term is nonpositive, the sum of the evaluated terms bounds a candidate's complete score from above, so a candidate whose bound falls below another's complete score is eliminated with its remaining evaluations, and execution stops when one fully scored candidate exceeds every remaining bound. The accuracy gain over MRNet comes from the decomposed scorer, not from selective execution; full and selective evaluation of the scorer make identical predictions on all 14,000 test instances, and selective execution removes 54.63% of the local candidate-unit evaluations while preserving them.

MNS. The original solver searches for rules, optimizes their weights, and selects the candidate with the smallest weighted residual. The executor instead verifies rule-expression pairs around the query in energy order and stops once the accepted rules yield a unique winner, after which the weight optimization is unnecessary; this rule is operational rather than exact (Sec. B.5). On the evaluation set the weight optimization shrinks from 5,600 Adam updates to one, at unchanged accuracy.

Ablation. Tab. 3 separates termination from the ordering that reaches it. Without termination there is no saving, whatever the order. On AlphaGeometry the three orderings save almost the same amount, so the task structure rather than the order decides what can be omitted. On RAVEN-FAIR the order matters. The learned energy raises the work reduction from 30.34% under random ordering to 54.63%, slightly above the 52.37% of a hand-designed heuristic, which is marginally faster in wall-clock terms because it costs nothing to compute. Termination is thus the general mechanism, and learned ordering pays off where the candidate computations leave real freedom in their order.

6 Conclusion

Answer determination separates the requested answer from the richer results a solver recovers; computation spent past that point is over-solving. Since possible answers can contract in any monotone pattern, no task-independent rule says what to skip, so we stop execution with a task-specific completion rule and order it with a learned answer-relevance energy. On five tasks this omits a third to nearly all of the task-specific computation at unchanged or better accuracy. Deriving completion rules automatically is the main open problem (Sec. F).

AI USE STATEMENT

Large language models were used to polish the writing and to check formatting. All ideas, proofs, code, and experiments are the authors' own, and the authors take full responsibility for the content.

ETHICS STATEMENT

All authors have read and adhere to the ICLR Code of Ethics. This work studies the efficiency of reasoning on synthetic and public benchmarks and raises no concerns regarding human subjects, privacy, or potentially harmful applications.

REPRODUCIBILITY STATEMENT

Benchmark descriptions are given in Sec. A, the task-specific instantiations with their completion rules, learned priorities, continuation procedures, and work metrics in Sec. B, implementation, training, and timing details in Sec. C, additional results in Sec. D, and the proof of Theorem 3.5 in Sec. E. Code and trained models will be released upon publication.

REFERENCES

Pranjal Aggarwal, Aman Madaan, Yiming Yang, and Mausam. Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with LLMs. In *Annual Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Daman Arora and Andrea Zanette. Training language models to reason efficiently. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Yaniv Benny, Niv Pekar, and Lior Wolf. Scale-localized abstract reasoning. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Do not think that much for 2+3=? on the overthinking of o1-like llms. *arXiv preprint arXiv:2412.21187*, 2024.

Kevin Ellis, Catherine Wong, Maxwell I. Nye, Mathias Sablé-Meyer, Lucas Morales, Luke B. Hewitt, Luc Cary, Armando Solar-Lezama, and Joshua B. Tenenbaum. DreamCoder: Bootstrapping inductive program synthesis with wake-sleep library learning. In *ACM SIGPLAN International Conference on Programming Language Design and Implementation (PLDI)*, 2021.

Gongfan Fang, Xinyin Ma, and Xinchao Wang. Thinkless: LLM learns when to think. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 2025.

Nicholas Hay, Stuart Russell, David Tolpin, and Solomon Eyal Shimony. Selecting computations: Theory and applications. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2012.

Tomasz Imieliński and Witold Lipski. Incomplete information in relational databases. *Journal of the ACM*, 31(4):761–791, 1984.

Shalmali Joshi, Junzhe Zhang, and Elias Bareinboim. Towards safe policy learning under partial identifiability: A causal approach. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, 2024.

Yang Li and Junier Oliva. Active feature acquisition with generative surrogate models. In *Proceedings of International Conference on Machine Learning (ICML)*, 2021.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2024.

Xin Liu and Lu Wang. Answer convergence as a signal for early stopping in reasoning. In *Annual Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2025.

Pasquale Minervini, Sebastian Riedel, Pontus Stenetorp, Edward Grefenstette, and Tim Rocktäschel. Learning reasoning strategies in end-to-end differentiable proving. In *Proceedings of International Conference on Machine Learning (ICML)*, 2020.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. s1: Simple test-time scaling. In *Annual Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2025.

Alexander Luke Ian Norcliffe, Changhee Lee, Fergus Imrie, Mihaela Van Der Schaar, and Pietro Lio. Stochastic encodings for active feature acquisition. In *Proceedings of International Conference on Machine Learning (ICML)*, 2025.

Maxwell I. Nye, Armando Solar-Lezama, Joshua B. Tenenbaum, and Brenden M. Lake. Learning compositional rules via neural program synthesis. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Rui Pan, Yinwei Dai, Zhihao Zhang, Gabriele Oliaro, Zhihao Jia, and Ravi Netravali. SpecReason: Fast and accurate inference-time compute via speculative reasoning. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Stuart Russell and Eric Wefald. Principles of metareasoning. *Artificial Intelligence*, 1991.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. CODI: Compressing chain-of-thought into continuous space via self-distillation. In *Annual Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2025.

Koustuv Sinha, Shagun Sodhani, Jin Dong, Joelle Pineau, and William L. Hamilton. CLUTRR: A diagnostic benchmark for inductive reasoning from text. In *Conference on Empirical Methods in Natural Language Processing and International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2025.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Na Zou, Hanjie Chen, and Xia Hu. Stop overthinking: A survey on efficient reasoning for large language models. *Transactions on Machine Learning Research*, 2025.

Trieu H. Trinh, Yuhuai Wu, Quoc V. Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 2024.

Jimmy Wang, Thomas P. Zollo, Richard Zemel, and Hongseok Namkoong. Adaptive elicitation of latent information using natural language. In *Proceedings of International Conference on Machine Learning (ICML)*, 2025.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for LLM problem-solving. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2025.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Zishun Yu, Tengyu Xu, Di Jin, Karthik Abinav Sankararaman, Yun He, Wenxuan Zhou, Zhouhao Zeng, Eryk Helenowski, Chen Zhu, Sinong Wang, Hao Ma, and Han Fang. Think smarter not harder: Adaptive reasoning with inference aware optimization. In *Proceedings of International Conference on Machine Learning (ICML)*, 2025.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. STaR: Bootstrapping reasoning with reasoning. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Chi Zhang, Baoxiong Jia, Mark Edmonds, Song-Chun Zhu, and Yixin Zhu. ACRE: Abstract causal REasoning beyond covariation. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

Wenhe Zhang, Chi Zhang, Yixin Zhu, and Song-Chun Zhu. Machine number sense: A dataset of visual arithmetic problems for abstract and relational reasoning. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

Zhiheng Zhang. Tight partial identification of causal effects with marginal distribution of unmeasured confounders. In *Proceedings of International Conference on Machine Learning (ICML)*, 2024.

Zhanke Zhou, Xiao Feng, Zhaocheng Zhu, Jiangchao Yao, Sanmi Koyejo, and Bo Han. From passive to active reasoning: Can large language models ask the right questions under incomplete information? In *Proceedings of International Conference on Machine Learning (ICML)*, 2025.

A Benchmark Descriptions

ACRE. ACRE (Zhang et al., 2021) studies abstract causal induction with a blicket machine. Given observations of object combinations and the resulting machine states, a query asks whether a new combination must activate the machine, must leave it inactive, or remains undetermined. We evaluate on 384 queries pooled over the independent and identically distributed (IID), compositional, and systematic splits, with 140 activated, 138 inactivated, and 106 undetermined answers.

CLUTRR. CLUTRR (Sinha et al., 2019) tests systematic generalization in relational reasoning. Given family relations among a set of people, a query asks for the kinship relation between two of them. We use the relation graph rather than the narrative as input and evaluate on 1,048 test queries with chain lengths from 2 to 10 (Tab. A3).

AlphaGeometry. AlphaGeometry (Trinh et al., 2024) pairs a language model that proposes auxiliary constructions with the symbolic deduction engine DDAR. Each problem provides point constructions and known geometric relations together with a target proposition to prove. We evaluate the DDAR component and our executor on a fixed 51-problem test split of the official JGEX-AG-231 benchmark. We construct a deterministic hash-based split from the parsed formal statements, assigning the 231 problems to training, validation, and test sets of sizes 135, 45, and 51, respectively. The split is independent of whether DDAR solves an instance. Neither the DDAR baseline nor our executor invokes AlphaGeometry's language model or generates language-model-proposed auxiliary constructions during evaluation.

RAVEN-FAIR. RAVEN-FAIR (Benny et al., 2021) is a bias-controlled variant of Raven's Progressive Matrices. The solver selects the missing panel of a 3×3 matrix from eight candidates using patterns in attributes such as number, position, shape, and color. We evaluate on the seven standard configurations, 2,000 test problems each.

MNS. Machine Number Sense (Zhang et al., 2020) consists of visual arithmetic problems in geometric layouts. Each problem embeds numbers in a geometric context, and the solver must infer the underlying arithmetic rule and fill in the missing value from the candidates. We evaluate on the evaluation split of the benchmark.

B Task-Specific Instantiations

Each subsection names the baseline and the computation it over-solves, states the completion rule and whether it is exact, describes the candidate computations that the energy $E_{\theta,b}$ orders and the supervision it is trained from, and defines the continuation procedure and the work metric behind Tab. 2.

B.1 ACRE

Baseline and over-solving. Under the OR mechanism, the machine activates whenever at least one activating object is present. NS-Opt fits a causal model to each problem and determines the queried state by continuous optimization over that model, thereby resolving the role of every object even when a single observation already fixes the query.

Completion rule. Let C_i and b_i be the object set and the machine state of observation i, and let T be the queried set. Define

$$Z = \bigcup_{i:b_i=0} C_i, \qquad R_i = C_i \backslash Z \quad (b_i = 1). \tag{A1}$$

Every object in Z is non-activating, and each R_i contains at least one activating object. For consistent observations, the possible machine states are $\{0\}$ if $T \subseteq Z$, $\{1\}$ if $R_i \subseteq T$ for some i, and $\{0, 1\}$ otherwise; the three cases are ACRE's *inactivated*, *activated*, and *undetermined* labels. Soundness of the first two cases follows from the definitions of Z and R_i. Completeness of the third holds because, when neither condition applies, declaring every object outside $Z \cup T$ activating gives an

explanation consistent with all observations under which T leaves the machine off, and declaring every object outside Z activating gives one under which T turns it on. The rule is therefore exact, and complete once every observation has been processed. The executor maintains Z and the sets R_i over the processed observations and stops as soon as one of the first two conditions holds for the query.

Learned priority. The energy ranks the unprocessed observations given the query and the current sets Z and R_i. Supervision marks the observations whose processing establishes one of the two conditions for the query, and the energy is trained to rank them first.

Continuation and work metric. Once every observation has been processed without either condition holding, the executor returns *undetermined*, which is the correct label in that case, and no other procedure is invoked. Work counts processed observation checks relative to processing all observations.

B.2 CLUTRR

Baseline and over-solving. CTP performs a multistep neural proof search separately for each candidate relation, searching the graph for every candidate even when a single path settles the query.

Completion rule. Kinship relations compose along a path. A composition table read off the training proofs maps a pair of relations to the set of relations it can entail, so that an ambiguous composition is represented by a set with several members. The executor maintains a chart of entailed relation sets over the intervals of the query paths and extends it one composition step at a time. The query is determined when the root interval of a complete path between the queried entities carries a single relation. Given the table, the rule is exact, since a single entailed relation at the root leaves no compatible alternative.

Learned priority. The energy ranks the executable composition steps from the relation pair, the path and interval context, the current chart, the answer state, and the graph context around the queried entities. Supervision traces each training proof back from the queried relation and marks the compositions that contribute to it. After each composition the chart is updated and the remaining steps are ranked again.

Continuation and work metric. Queries that remain open after the compositions are passed to CTP for a complete proof search. Work counts executed composition-table lookups relative to complete composition.

B.3 AlphaGeometry

Baseline and over-solving. DDAR, the symbolic engine of AlphaGeometry, repeatedly matches applicable geometric rules, enumerates bindings, and expands the proof state until the target is established, deriving many intermediate relations that the target does not need.

Completion rule. The target proposition is established from the current proof state either by a geometric rule application or directly from the existing algebraic relations. The executor checks the target after every update and terminates as soon as it is derived, tracing the derivation through DDAR's dependency graph to recover the proof. A derivation is a proof, so the rule is exact.

Learned priority. The energy combines target-rule associations with a state-dependent ordering over the computations compatible with the target. Supervision from training proofs and execution rollouts favors the computations that establish the target or reach it at lower search cost, and the energy orders geometric rule applications and algebraic checks as the proof state evolves.

Continuation and work metric. Unresolved states continue with unrestricted DDAR from the current proof state. Work counts geometric predicate checks relative to a full DDAR run.

B.4 RAVEN-FAIR

Baseline and over-solving. MRNet scores all candidates with an integrated multiscale relational model, so every candidate receives the full computation even when a few local comparisons already rule it out. Our executor uses a separate decomposed scorer, described next, whose accuracy under full evaluation is reported in Sec. D.

Completion rule. We define local units $u \in \mathcal{U}$ by component and attribute, learn their rule responses, and write the score of candidate c as

$$s(c) = \sum_{u \in \mathcal{U}} s_{u,c}, \qquad s_{u,c} \leqslant 0. \tag{A2}$$

For a candidate whose evaluated units form $\mathcal{U}_c \subseteq \mathcal{U}$, the partial sum

$$\bar{s}(c) = \sum_{u \in \mathcal{U}_c} s_{u,c} \geqslant s(c)$$

bounds its complete score from above. A candidate is eliminated, together with its remaining unit evaluations, once its bound falls below the complete score of another candidate, and the executor returns a fully evaluated candidate $c^\star$ once $s(c^\star) > \bar{s}(c)$ for every remaining candidate c. In the formalism of Sec. 3, Ω is the set of completions of the unevaluated terms and a the induced selection, and the rule is exact for the decomposed scorer. When multiple answer candidates have the same completed score, we select the candidate with the smallest index. A candidate whose upper bound ties the incumbent is retained until this tie-breaking rule can be applied.

Learned priority. The energy ranks the remaining local units from the current partial scores and the candidate-unit context. Supervision from execution rollouts favors the units whose evaluation leads to lower remaining evaluation cost. After each unit evaluation the bounds are updated and the remaining units are ranked again.

Continuation and work metric. Candidates that remain unresolved when the budget is exhausted are completed by evaluating their remaining units. Work counts executed candidate-unit evaluations relative to full evaluation.

B.5 MNS

Baseline and over-solving. The original solver searches for candidate rules, optimizes their weights, and selects the candidate with the smallest weighted residual

$$J(a) = \sum_i \lambda_i h_i(a), \tag{A3}$$

where $h_i(a)$ is the residual of candidate a under rule i and $\lambda_i \geqslant 0$. It continues the rule search and the weight optimization after a small set of rules and expressions already determines the query.

Completion rule. The executor verifies candidate rules and their expressions around the query in ranked order and adds each accepted rule, with its residual scores over the answer candidates, to the current answer state. The query is declared determined once the accepted rules agree on a single winner, a minimum number of rules have been accepted, and the winner's margin over the runner-up exceeds a threshold. This rule is operational rather than exact, since agreement among the accepted rules does not exclude a rule not yet examined.

Learned priority. The energy, a small network we call TargetReasoner, ranks rule-expression pairs from the visible rows, the query position, and the representation of the candidate rule. Supervision from a teacher run marks the pairs that support the answer. The executor verifies the pairs in ranked order and updates the answer state as rules are accepted.

Continuation and work metric. Queries that remain open after ranked verification continue with the original solver restricted to the query. Work counts weight-optimization steps of the original solver's minimax-entropy procedure.

Table A1: **Implementation summary.** Each task learns a lightweight energy over its candidate computations.

Task	Energy model	Supervision	Training	Inference
ACRE	MLP, 2.1K parameters	Reachable states	Adam, 3×10^{-3}, 12 epochs	Completion-driven
CLUTRR	Linear, 218 parameters	1,464 compositions	Logistic regression	Completion-driven
AlphaGeometry	ExtraTrees and linear	134 traces and 268 rollouts	Pairwise ranking	Target probe, 4 ms
RAVEN-FAIR	MLP, 481 parameters	8,192 rollout states	AdamW, 2×10^{-3}, 80 epochs	Bound-driven
MNS	TargetReasoner, 44.4K parameters	Teacher rule-expression pairs	AdamW, 2×10^{-3}, 12 epochs	Gate 0.5, top 16

Table A2: **ACRE accuracy by answer category.** Both systems are evaluated on the same 384 queries pooled over the IID, compositional, and systematic splits.

Answer	n	NS-Opt	Ours
Activated	140	88.57%	100.00%
Inactivated	138	93.48%	100.00%
Undetermined	106	11.32%	100.00%

C Implementation Details

Tab. A1 summarizes the energy model, the supervision, and the training and inference settings of each task.

Timing and baselines. Within each benchmark, our executor and its baseline are timed on the same hardware, an AMD Ryzen 7 9800X3D CPU and an NVIDIA GeForce RTX 5070 Ti GPU, under matched execution settings. ACRE uses five repeated calls per problem, CLUTRR three complete evaluation passes, AlphaGeometry three repeated runs on the same 51 problems, RAVEN-FAIR three interleaved runs on 1,024 fixed instances, and MNS one complete traversal of the evaluation set. Timing covers online solving and control and excludes offline training and shared data loading. ACRE, CLUTRR, AlphaGeometry, and RAVEN-FAIR use local reruns of the released baseline implementations; MNS uses the original solver. For ACRE, both systems operate directly on the symbolic object sets and machine states used by the reasoning stage, without the visual perception frontend. For RAVEN-FAIR, our executor uses the same visual panel input modality as the baseline. For AlphaGeometry, both the DDAR baseline and our method operate on the supplied formal constructions and goal. Neither invokes AlphaGeometry's language model nor generates language-model-proposed auxiliary points during evaluation; our learned priority model is a separately trained rule ranker.

D Additional Results

ACRE. Tab. A2 breaks the ACRE results down by answer category. The improvement holds on each of the three splits, and the largest gap is on *undetermined* queries, where NS-Opt reaches 11.32% and the executor answers all 106 correctly. On *activated* and *inactivated* queries NS-Opt reaches 88.57% and 93.48%, and the executor 100% on both.

CLUTRR. Tab. A3 reports accuracy and work reduction by test chain length k. At $k = 2$ both systems reach 100% and perform the same compositions. The savings begin at $k = 3$ and grow with chain length to about 75% on the longest chains. The accuracy gap opens at $k = 4$, where CTP falls to 62.34% while the executor stays at 84.42%, and for $k \geqslant 5$ the executor stays above 93% while CTP declines to 62.30% at $k = 10$.

AlphaGeometry. On the 51 held-out JGEX-AG-231 problems described in Sec. A, DDAR and our executor solve exactly the same 45 problems, with the remaining six ending in the same *saturated* state. Across the 51 problems, DDAR executes 3,018,389 geometric predicate checks, while our executor executes 1,993,638. Selective execution removes more than one million predicate checks while preserving the same solved set.

Table A3: **CLUTRR results by test chain length.** Work reduction counts composition-table lookups relative to complete composition.

k	n	CTP Acc.	Ours Acc.	Work Red.
2	38	100.00%	100.00%	0.00%
3	107	90.65%	90.65%	43.98%
4	77	62.34%	84.42%	53.08%
5	185	83.78%	98.38%	65.92%
6	105	83.81%	97.14%	69.87%
7	155	75.48%	98.06%	74.21%
8	135	72.59%	96.30%	75.15%
9	124	66.13%	95.97%	75.74%
10	122	62.30%	93.44%	74.73%

Table A4: **RAVEN-FAIR results by configuration.** Full denotes complete evaluation of the decomposed scorer and Selective its evaluation under the executor; work reduction counts candidate-unit evaluations relative to Full.

Configuration	MRNet	Full	Selective	Work Red.
Center	96.80%	100.00%	100.00%	56.75%
2×2 Grid	74.45%	99.70%	99.70%	57.40%
3×3 Grid	69.95%	98.15%	98.15%	56.22%
L-R	99.00%	99.95%	99.95%	53.38%
U-D	98.60%	100.00%	100.00%	52.39%
O-IC	98.15%	100.00%	100.00%	53.76%
O-IG	74.60%	99.35%	99.35%	52.54%

RAVEN-FAIR. Tab. A4 reports accuracy by configuration for MRNet, for the decomposed scorer under full evaluation, and for selective execution. Full and selective evaluation agree on all 14,000 test problems, and selective execution removes 52.39% to 57.40% of the candidate-unit evaluations depending on the configuration.

MNS. Over the evaluation set, the weight optimization of the original solver runs 5,600 Adam updates with 560,000 Langevin steps; under selective execution it runs one Adam update with 100 Langevin steps, which is the 99.98% reduction reported in Tab. 2, at an unchanged accuracy of 75.00%.

E Proof of the Characterization Theorem

Proof of Theorem 3.5. Necessity follows from Proposition 3.4 and $\Omega_I = \{z^\star\}$.

For sufficiency, let $G(I) = \{y^\star\}$, and let the true explanation $z^\star$ have value 0 at every information item, with answer $y^\star$. For each $A \subsetneq I$ and each $y \in G(A)\backslash\{y^\star\}$, construct an explanation

$$z_i^{A,y} = \begin{cases} 0, & i \in A, \\ u_{A,y}, & i \notin A, \end{cases}$$

where the symbols $u_{A,y}$ are distinct from one another and from 0, and set $a(z^{A,y}) = y$. Let Ω consist of the true explanation and all explanations constructed above. Each $z^{A,y}$ agrees with the true explanation exactly on A, so

$$z^{A,y} \in \Omega_S \quad \Longleftrightarrow \quad S \subseteq A.$$

Consequently, for every $S \subseteq I$,

$$F(S) = \{y^\star\} \cup \bigcup_{S \subseteq A \subsetneq I} \big(G(A)\backslash\{y^\star\}\big) = G(S).$$

The final equality follows from monotonicity, from $y^\star \in G(S)$, and from the inclusion of the term $A = S$ in the union when $S \subsetneq I$. When $S = I$, both sides equal $\{y^\star\}$. □

F LIMITATIONS AND FUTURE WORK

The completion rules are derived by hand from each task's semantics. Deriving them automatically, or learning approximate rules with calibrated soundness, would extend the approach to tasks without closed-form structure. The energy is trained per task from task-specific supervision, and we have not studied transfer across tasks or the sensitivity to the budget B beyond the settings reported. The theory characterizes what can happen to the possible answers but not what typically happens; a probabilistic version that models the expected contraction under a task distribution is a natural next step.