Hierarchical Models of Multi-Agent Systems: Strategic Ability and Model Checking

Rustam Galimullin[1] · Wojtek Jamroga[2] · Damian Kurpiewski[2] · Vadim Malvone[3] · **Aniello Murano[4]**

[1]University of Bergen · [2]Polish Academy of Sciences & NCU Toruń · [3]Télécom Paris (IP Paris) · [4]University of Naples Federico II

KR 2026
Lisbon, Portugal
22 July 2026

MOTIVATION

Why Hierarchy?

Strategic reasoning does not scale well.
As agents and their interactions multiply, checking what coalitions of agents can achieve quickly becomes intractable.

Real systems are built from components.
The same subsystem shows up repeatedly, in various contexts.

Hierarchy is the natural abstraction.
A coarse top level, with selected nodes refined into detailed sub-models — succinct and modular.

Examples

Authorisation module
Specified once — reused as a building block of many different e-voting protocols.

Autonomous robot fleets
Different robots share the very same modules, such as motor, vision and communication.

Goal — reason about strategic ability while keeping the hierarchy.

BACKGROUND

Concurrent Game Models (CGMs)

Agents act simultaneously

At every step, each agent independently picks one of its available actions.

States, actions, availability

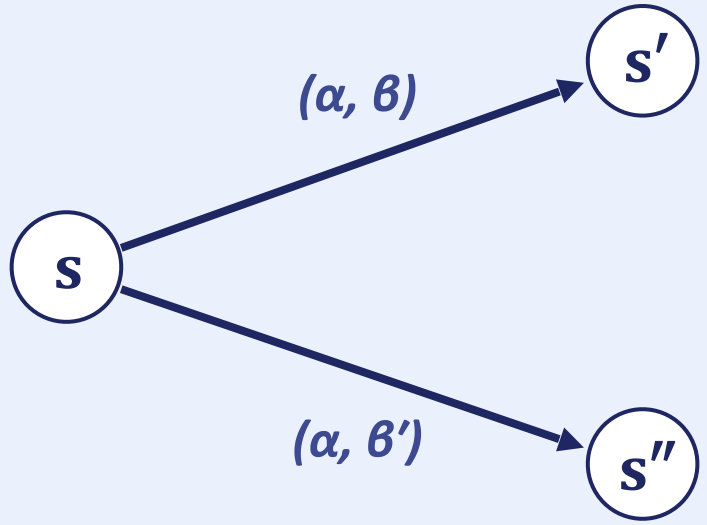
$\mathbf{act}(s,\ a)$ — the actions agent a may take in state s.

Joint transitions

$\mathbf{o}(s,\ \alpha_1,\ \ldots,\ \alpha_n)$ — the next state depends on everyone's choice.

Valuation

$V(s)$ — the atomic facts that hold in each state.



Two agents: the outcome of playing α depends on the other agent's simultaneous choice.

Paths $\pi = s_0\ s_1\ s_2 \ldots$ arise from repeated joint actions — strategies will pick actions along them.

BACKGROUND

Alternating-Time Temporal Logic (ATL)

The coalition operator $\langle\langle C \rangle\rangle$ — *"coalition C has a strategy to guarantee …"*

$\langle\langle C \rangle\rangle X\, \varphi$

NEXT

C can force φ to hold at the very next step.

$\langle\langle C \rangle\rangle \psi\, U\, \varphi$

UNTIL

C can maintain ψ until φ eventually becomes true.

$\langle\langle C \rangle\rangle \psi\, R\, \varphi$

RELEASE

C can keep φ true, unless ψ releases the requirement.

Memoryless strategies

$\sigma : St \to Act$ — one action per state; a joint strategy gives one per coalition member.

Model checking

Given a model M, a state s and a formula φ: decide whether $M, s \models \varphi$.

Semantics: there exists a joint strategy for C such that on all compatible paths the objective holds.

STATE OF THE ART

What Was Missing

Hierarchical verification — reactive, single-agent systems

- Hierarchical state machines + CTL — well understood (Alur & Yannakakis 2001)
- Interface automata, hierarchical module checking, hierarchical cost-parity games, …
- A mature line of algorithms and tools

Strategic multi-agent systems … ?

- Agents act proactively — they pursue objectives, they don’t just react
- Objectives are strategic abilities: what can a coalition enforce?
- $\langle\langle C \rangle\rangle$ quantifies over strategies — plain temporal model checking does not apply

Our contribution — lift hierarchical verification to concurrent games and ATL.

THIS PAPER

Our Contribution

1

HCGMs

Hierarchical concurrent game models — modules embed sub-modules via boxes; components are written once and reused many times.

2

Semantics on hierarchies

A flattening construction unfolds an HCGM into an equivalent flat CGM — giving ATL a precise meaning on hierarchical models.

3

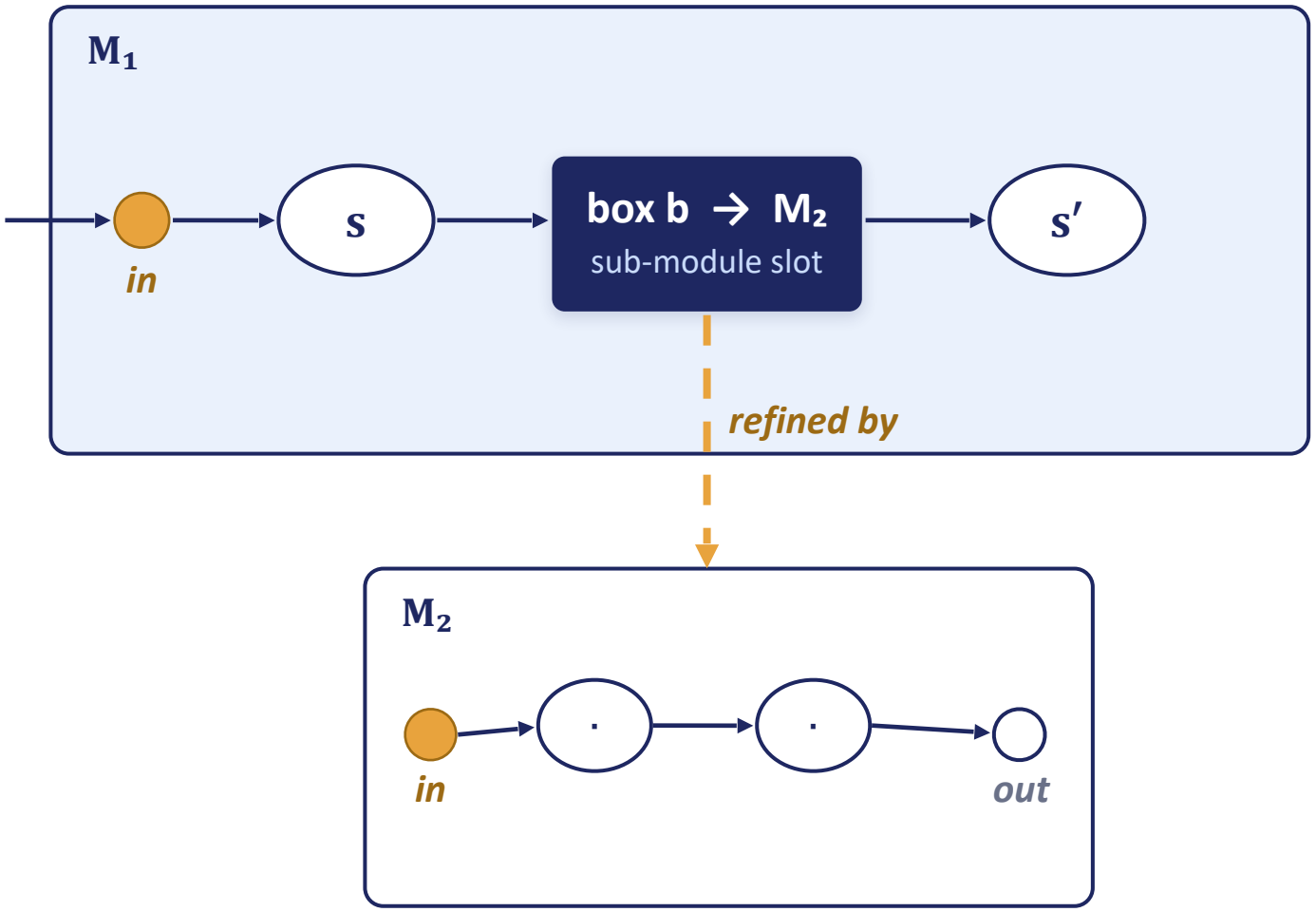
Direct model checking

An algorithm that runs on the hierarchy itself — PSPACE-complete in general; PTIME, even linear, for bounded exits and formulas.

Extends Alur & Yannakakis' hierarchical CTL results to strategically interacting agents — same complexity, richer framework.

THE MODEL

Hierarchical CGMs — The Idea



A box is a placeholder

It stands for a deeper module, plugged in at that point.

Strictly deeper, acyclic

Map sends boxes only to lower levels — no recursion, embedding is well-founded.

Self-contained top

The top module has no exits: it models the whole closed system.

Local agents & actions

Each module brings its own agents — the sets may differ across levels.

THE MODEL, FORMALLY

Inside a Module Mi

Agti — active agents of the module; agent sets may differ from module to module.

Sti, Ini, Outi — states, entry points, exits (mutually disjoint; the top module has no exits).

Bi — boxes: the slots where sub-modules are plugged in.

Mapi : Bi → deeper modules — which module fills each box (only strictly deeper ones).

acti — the actions available to each agent at every decision point.

oi — transitions: to a local state, into a box (via its inputs), or out of the module.

Vi — the atomic facts holding at states, entries and exits.

Key: the exits of a box behave like local states of the parent — an exit may even feed back into an entry.

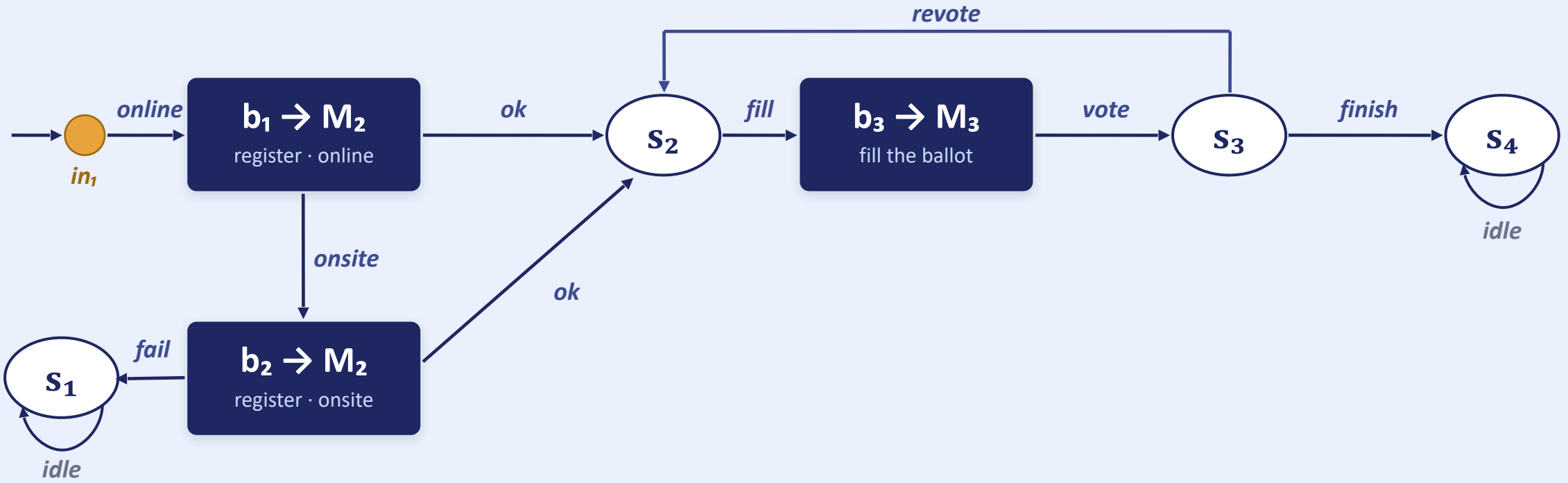
A hierarchical model is a sequence $M = (M_1, \ldots, M_k)$ *of such modules, with* M_1 *on top.*

THE MODEL AT WORK

Running Example — Hierarchical E-Voting

M_1 — top-level voting procedure

Agt_1 = { voter v, election authority ea }



M_2 (registration) and M_3 (ballot filling) are specified once — their internal structure stays hidden at this level.

Reuse
M_2 fills two boxes — written once, used twice.

Different agents
M_3 involves only the voter, election authority plays no role there.

Modularity
Components can be specified and verified independently.

FROM HIERARCHY TO CGM

Semantics — Flatten, Then Evaluate

Agents

Take the union of the agent sets over all modules.

States

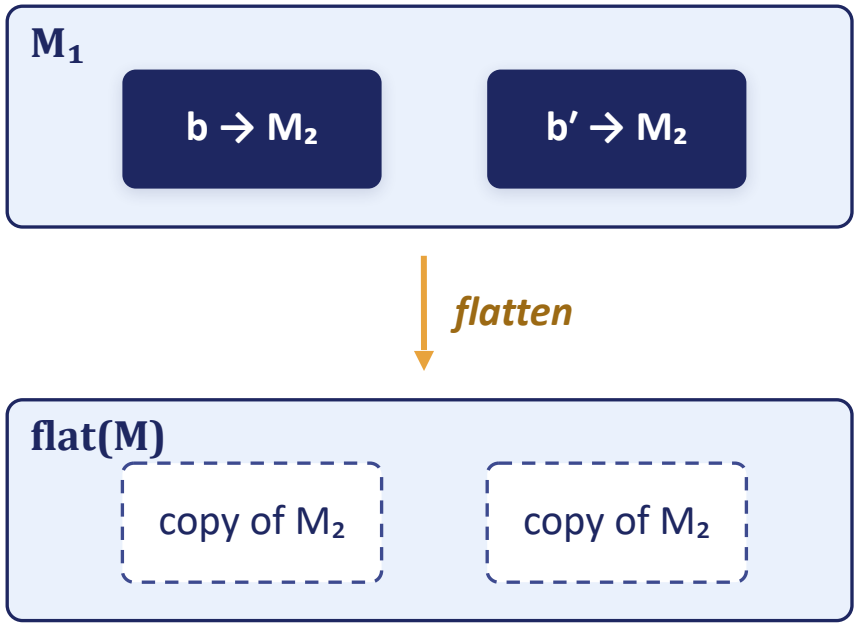
Recursive disjoint union — one fresh copy of a module for every occurrence of a box.

Actions

Union of all modules' actions, plus a fresh idle action.

Inactive agents idle

Agents absent from a module play idle and never influence the outcome there.



same module, two boxes → two inlined copies

ATL on an HCGM = ATL on flat(M) — hierarchy adds succinctness, not new truth.

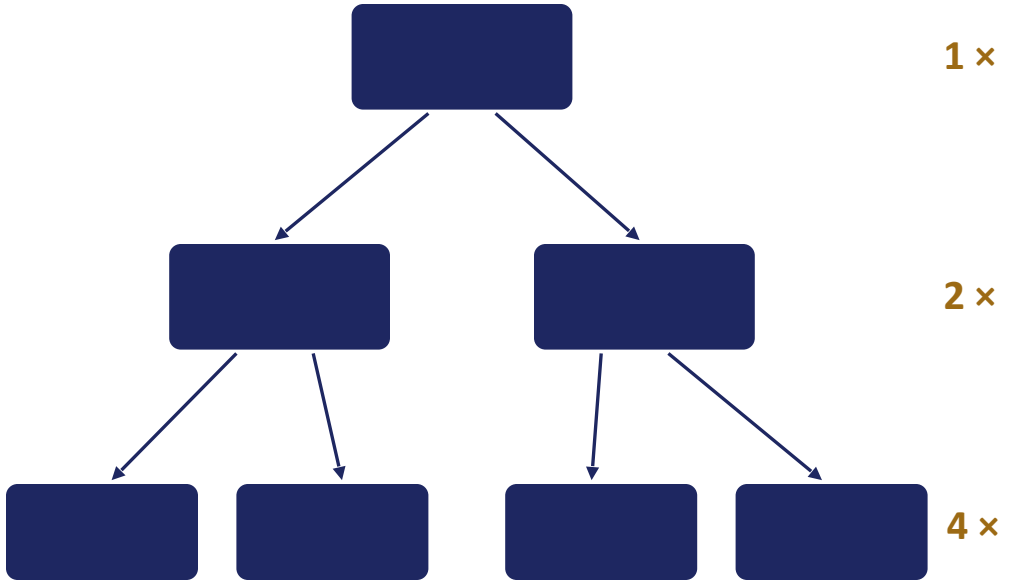
WHY NOT JUST FLATTEN?

The Price of Flattening

$$|\mathrm{flat}(M)| = O(|M|^{\mathrm{nd}(M)})$$

exponential in the nesting depth nd(M)

The same module may fill many boxes — so copies multiply, level after level.



copies of copies of copies …

Verification via flattening

$O(|\varphi| \cdot |M|^{\mathrm{nd}(M)})$ — in practice models are orders of magnitude larger than formulas.

THE PROBLEM

Model Checking — Two Routes

Route 1 — flatten & check

- Build flat(M) explicitly
- Run standard ATL model checking — polynomial in the (flat) model
- Total cost: $O(|\varphi| \cdot |M|^{nd(M)})$

Exponential in the nesting depth — impractical.

Route 2 (ours) — stay hierarchical

- Process modules bottom-up — never build flat(M)
- Local checks with the local coalition $C \cap Agt_i$
- A strategic generalisation of Alur & Yannakakis' CTL algorithm

Exponential only in $|\phi|$ and #exits — both small in practice.

Two obstacles for Route 2: (a) modules have different agent sets; (b) whether an exit satisfies a formula depends on the context above.

The Algorithm — Case $\langle\langle C \rangle\rangle X \chi$

1 **Bottom-up, per module**

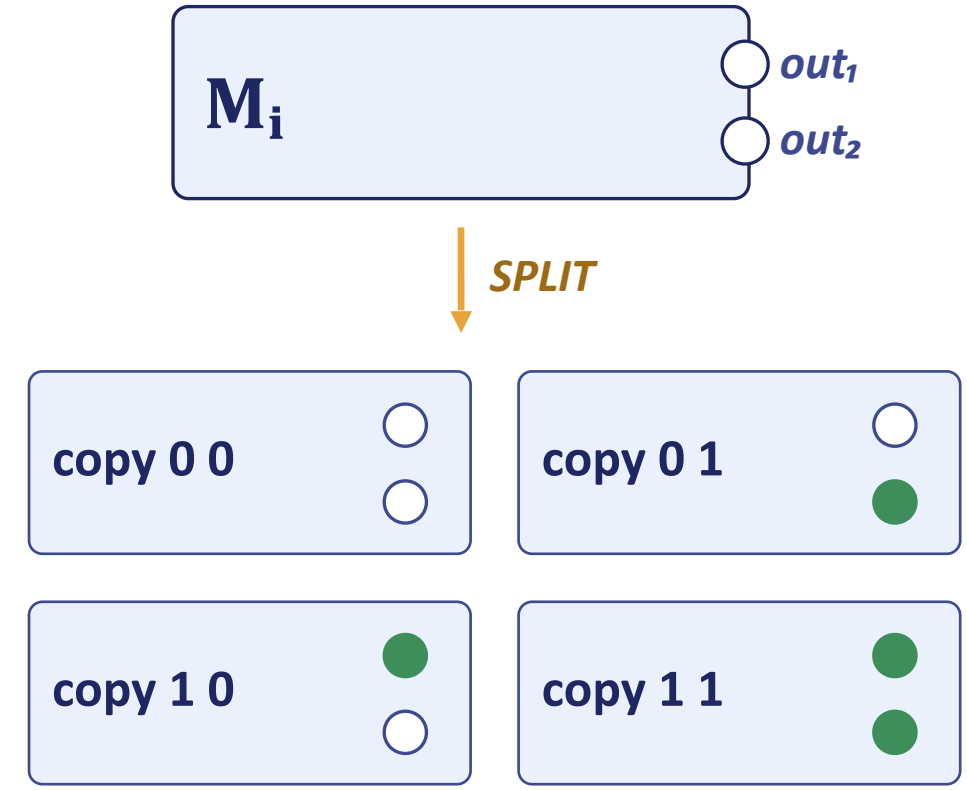
HCGM2CGM builds a partial local CGM — box interiors are resolved below, exits above.

2 **Check locally**

Standard ATL model checking with the local coalition $C \cap Agt_i$; agents outside the module idle (ε).

3 **SPLIT — keep every context**

An exit's truth depends on the parent. Keep $2^{\#exits}$ copies of the module — one per truth assignment on its exits — and re-map boxes to the matching copy.



one copy per labelling of the exits

green means exit satisfies the formula

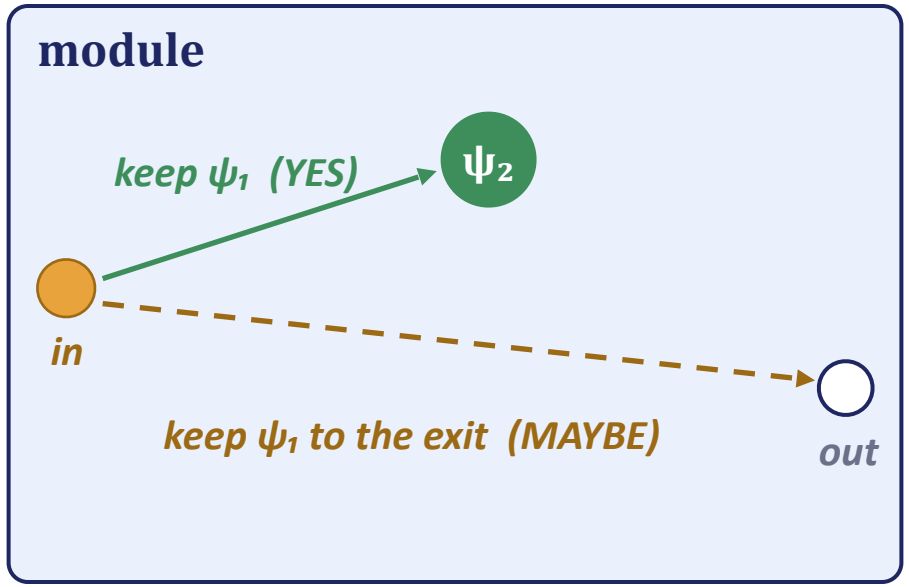
The Algorithm — Case $\langle\langle C \rangle\rangle \psi_1 \, U \, \psi_2$

SUMMARY

classify every input of every module, bottom-up:

YES	the coalition can reach ψ_2 inside the module, maintaining ψ_1 on the way
MAYBE	it can maintain ψ_1 all the way to an exit — ψ_2 may still come higher up
NO	neither — from this input the objective is hopeless

SPLIT with θ-formulas. Local ATL checks (θ_1, θ_2) decide each exit's status and re-map boxes accordingly.
Release is handled dually.



Interface summaries let parent modules reason without re-entering the child.

MAIN RESULTS

Complexity

THEOREM 1

PSPACE-complete

Model checking ATL for HCGMs, w.r.t. the sizes of the model and the formula — matching hierarchical CTL: the strategic dimension comes at no extra asymptotic cost.

THEOREM 2

PTIME — linear

For formulas of bounded length over modules with a bounded number of exits: P-complete, and solvable in linear time in the size of the HCGM.

Running time $O(|M| \cdot 2^{|\varphi|d})$ · polynomial space · d = max number of exits of a module

The exponent sits in the small parameters — the formula and the interface — not in the model.

DISCUSSION

Why This Wins in Practice

Formulas are short, models are huge.

Typical requirements use a handful of operators; models are orders of magnitude larger.

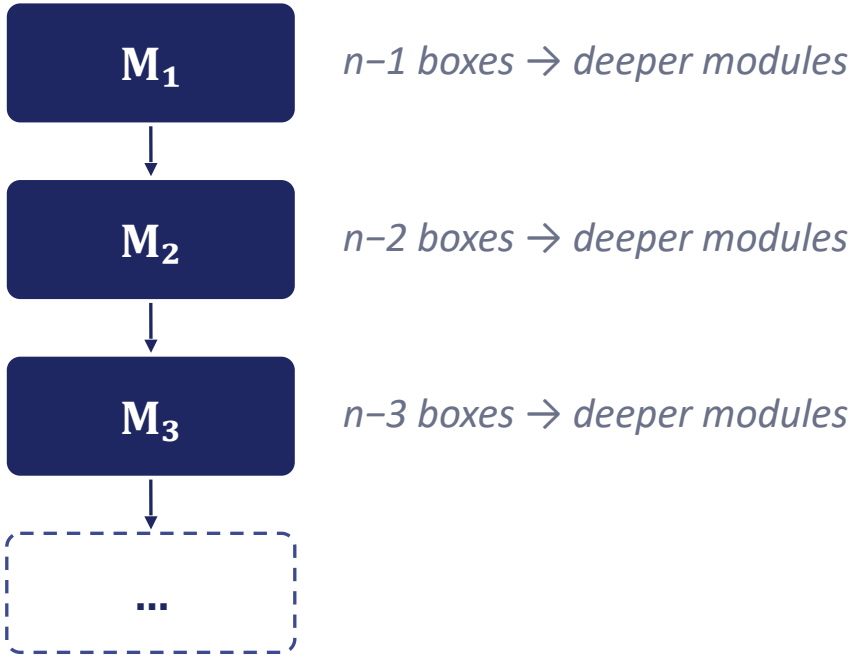
Boxes are function-like.

Good design keeps interfaces small: true/false, grant/deny, ack/nack/timeout — few exits by construction.

exponential → polynomial

There are HCGM families whose flattening has exponentially many states — while our algorithm still runs in polynomial time.

A hard family for flattening



flat(M): exponentially many states.
Hierarchical MC: still P.

WRAPPING UP

Conclusions & Road Ahead

WHAT WE DID

- HCGMs — hierarchical, reusable models of strategic multi-agent systems
- ATL semantics via flattening — succinctness without changing truth
- Direct model checking — PSPACE-complete; linear time for bounded exits & formulas

WHAT'S NEXT

- Implementation & benchmarks — extending MCMAS, STV, VITAMIN
- Abstraction techniques for HCGMs
- Richer logics — ATL*, Strategy Logic
- Imperfect information, perfect recall, asynchrony, module communication

Thank you — questions welcome!