Metodi formali per la verifica dell'affidabilità di sistemi: materiale didattico

Prima parte: verifica basata su logica temporale lineare

- Peled: Software Reliability Methods, cap. 1.
- Katoen: Concepts, Algorithms, and Tools for Model Checking, cap. 1 (escluso 1.4) e 2.
- Logica temporale e verifica di proprietà dei programmi (dispense) cialdea.dia.uniroma3.it/teaching/logica/materiale/dispense-LTL.pdf

Seconda parte: Modellazione e verifica di sistemi con proprietà temporali

- Katoen: Concepts, Algorithms, and Tools for Model Checking.
- Behrmann, David, Larsen: A Tutorial on Uppaal 4.0.

Verifica

Importanza della verifica di sistemi (safety-critical, commercially critical, mission critical): malfunzionamenti del sistema possono avere conseguenze disastrose (e estrememente costose).

- Il baco del 2000: gli US spesero più di 100.000 milioni di dollari per combattere il *Y2K-bug*. Non successe nulla di grave, ma ci furono molti problemi "minori" (sistemi, apparecchiature, carte di credito, ... che smisero di funzionare o non funzionarono come dovuto).
- Errore nella divisione floating-point nel processore Pentium Intel (1994): circa 500 milioni di dollari
- Errore (software) nel missile Ariane-5 (ESA, esploso nel 1996), del valore di 500 milioni di dollari

Sistemi SW controllano gran parte della nostra vita (telefoni, banche, aerei,...):

L'affidabilità dell'elaborazione delle informazioni è un punto chiave nel processo di progettazione del software

Sviluppo del software

Lo sviluppo del SW è cambiato

- coinvolge decine o centinaia di programmatori su uno stesso progetto
- migliaia/milioni di linee di codice
- persone dislocate

Difficoltà di controllare il processo di sviluppo del SW

$\Rightarrow$ metodologie di progettazione

$\Rightarrow$ metodi per identificare errori

Tecniche per la verifica e validazione del software (generalmente integrati nel processo di sviluppo del SW) e **complementari** tra loro:

- **Peer reviewing:** tecnica statica (simulazione), esame manuale del codice. Identifica tra il 31 e il 93% degli errori
- **Testing:** tecnica dinamica, si applica al sistema implementato, può dimostrare la presenza di errori, non la loro assenza.
- **Verifica formale:** per dimostrare che il sistema si comporta correttamente. Può applicarsi fin di primi stadi di sviluppo del progetto (per identificare gli errori prima possibile).

Metodi Formali per la verifica di sistemi

Collezione di notazioni e tecniche per descrivere a analizzare sistemi

Formali: basati su qualche teoria matematica (logica, automi, teoria dei grafi)

Verifica: applicazione di una tecnica manuale o automatica che possa aiutare a stabilire se un sistema soddisfa una data proprietà o si comporta in accordo a una specifica data.

Una tecnica di verifica formale include:

1. **Una metodologia per modellare sistemi:** rappresentare un sistema in termini di oggetti matematici, astraendo e semplificandone la descrizione
2. **Un linguaggio di specifica, per descrivere le proprietà del sistema modellato.** Si utilizzano formalismi logici, in genere una **logica temporale**.
3. **Un metodo di verifica:** tecniche di analisi formale per verificare se un sistema soddisfa la sua specifica.

Approcci fondamentali alla verifica formale: verifica deduttiva

Inizialmente con lo scopo di assicurare la correttezza di un programma.

Il sistema è descritto da un insieme di formule S (in una logica adatta allo scopo), la specifica è una formula A. Il metodo di verifica consiste nel dimostrare che $S \vdash A$.

Normalmente non sono metodi completamente automatizzabili.

Influenza sullo sviluppo del SW: nozione di invariante.

Approcci fondamentali alla verifica formale: model checking

Il sistema è rappresentato da una struttura matematica $\mathcal{M}$ (rappresentabile in modo finito), che corrisponde a un'interpretazione o un insieme di interpretazioni di una logica (adatta allo scopo). La specifica è una formula A. Il metodo di verifica consiste nel dimostrare che $\mathcal{M} \models A$.

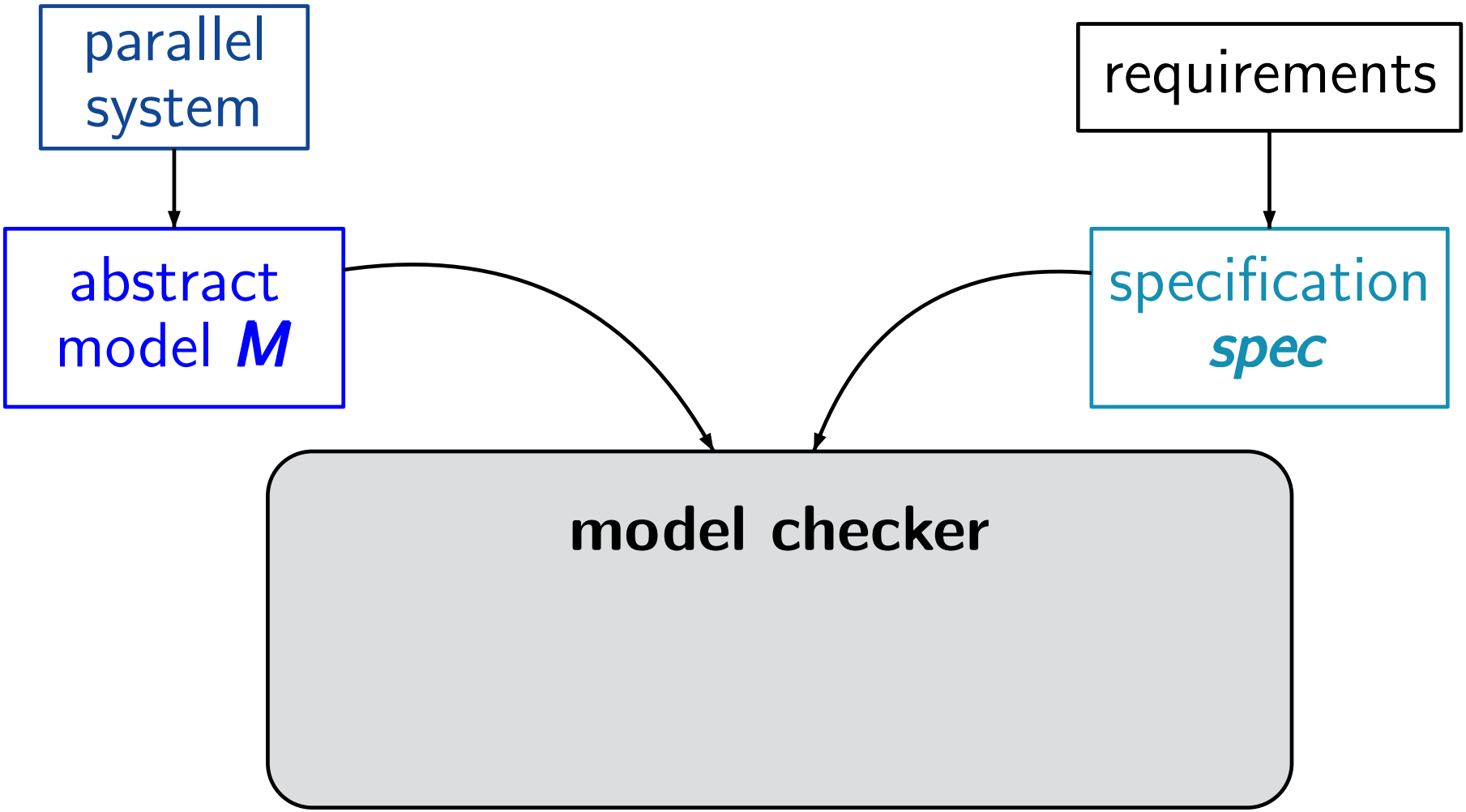
Il metodo di verifica è generalmente automatizzabile.

Applicabile a sistemi a stati finiti (dove l'insieme di valori che può assumere ciascuna variabile è finito).

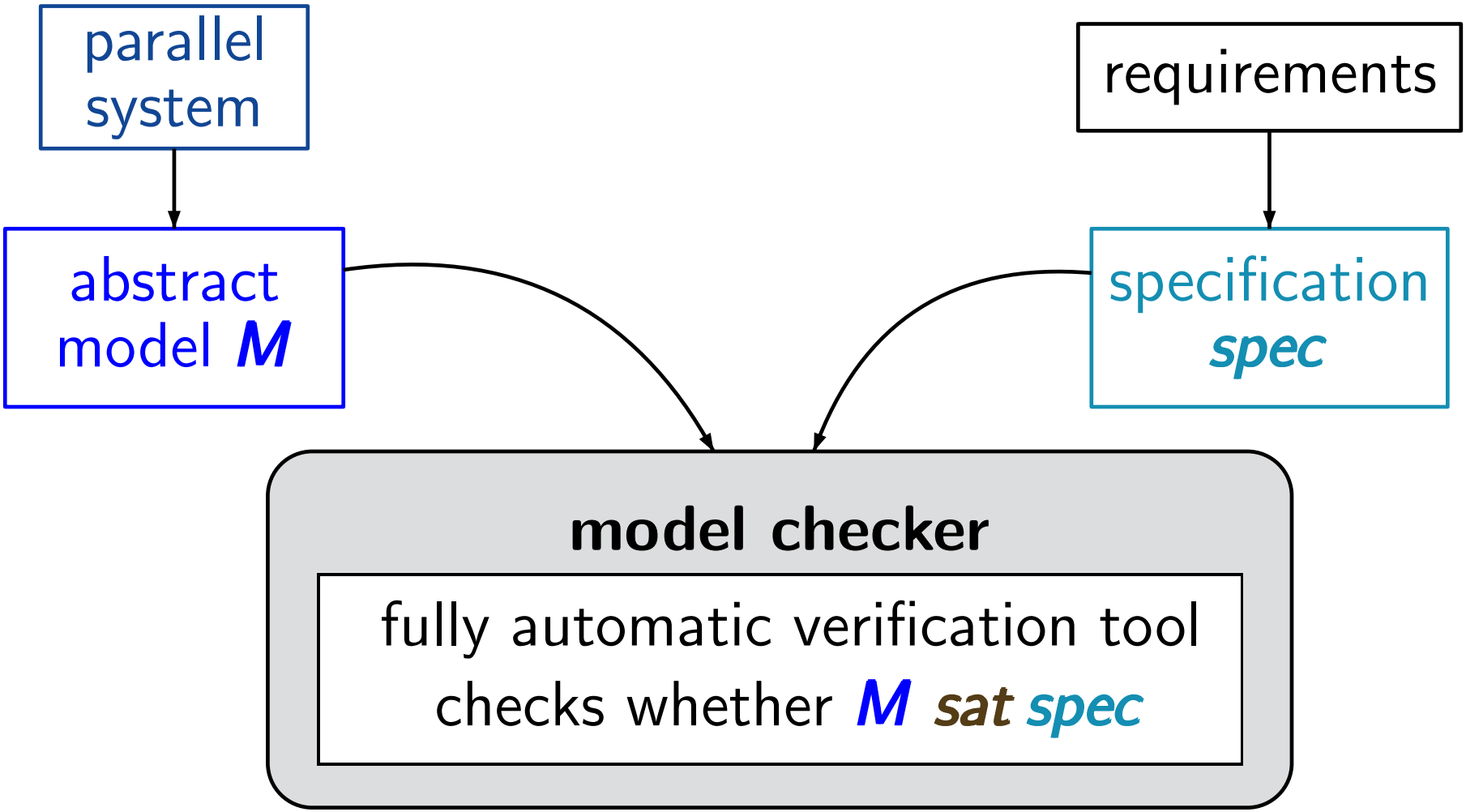
Metodi adatti a verificare protocolli di comunicazione, sistemi concorrenti, o rappresentazioni astratte di algoritmi.

L'applicazione di metodi formali di verifica è oggi spesso **integrata** nel processo di sviluppo del SW: metodologia di sviluppo per raffinamenti (specifica e test), conservando la correttezza da uno stadio dello sviluppo al successivo.

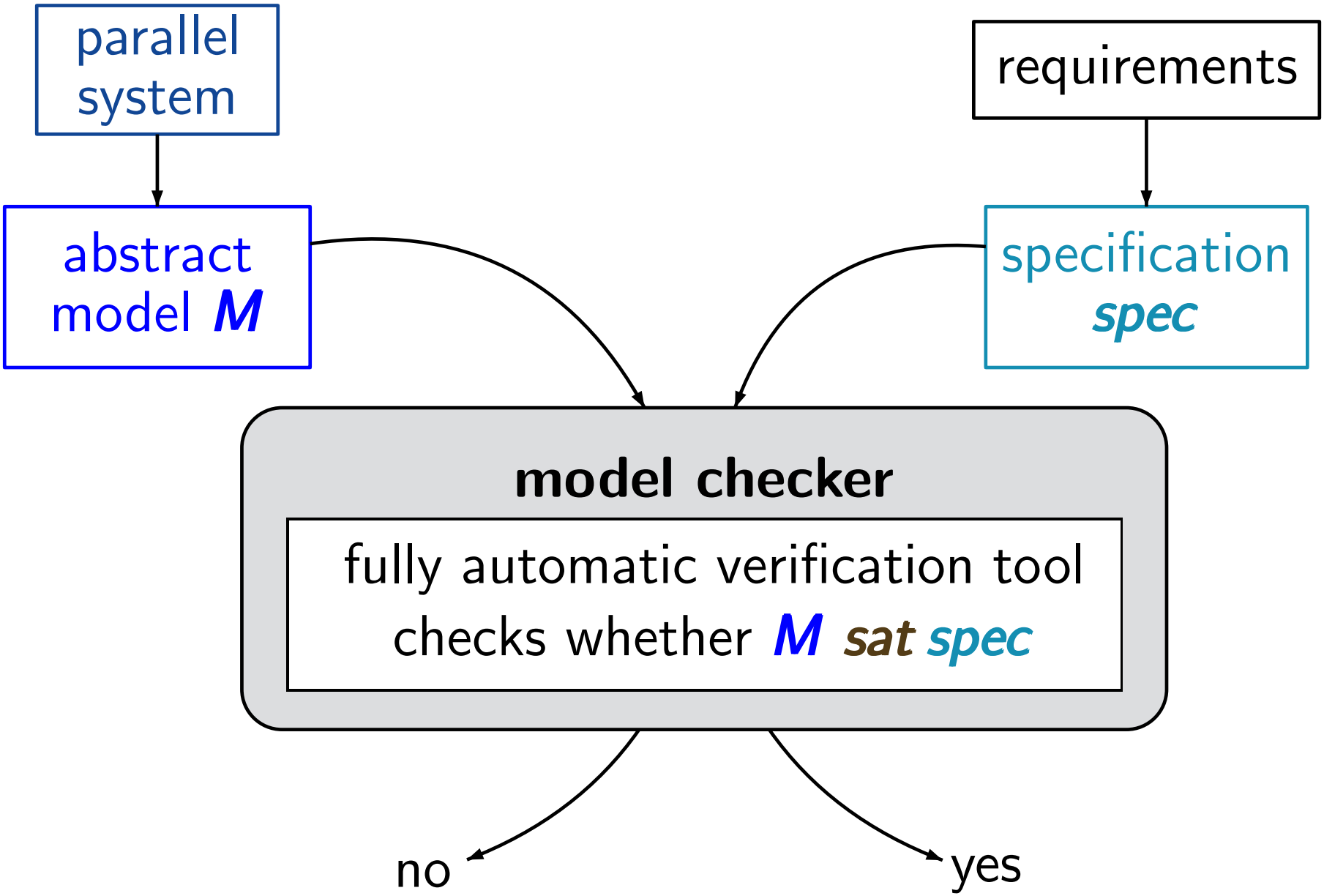
Model checking



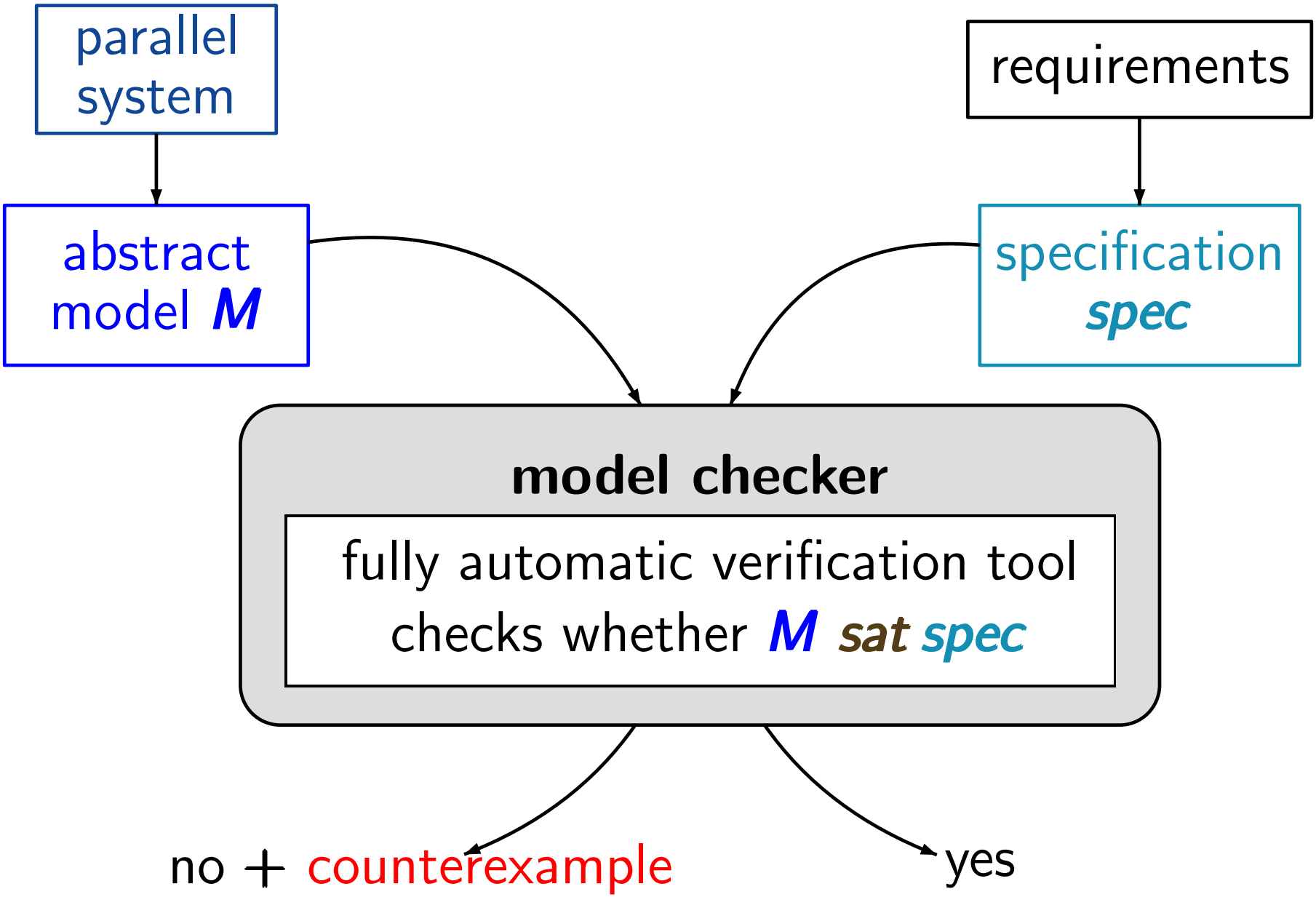
Model checking



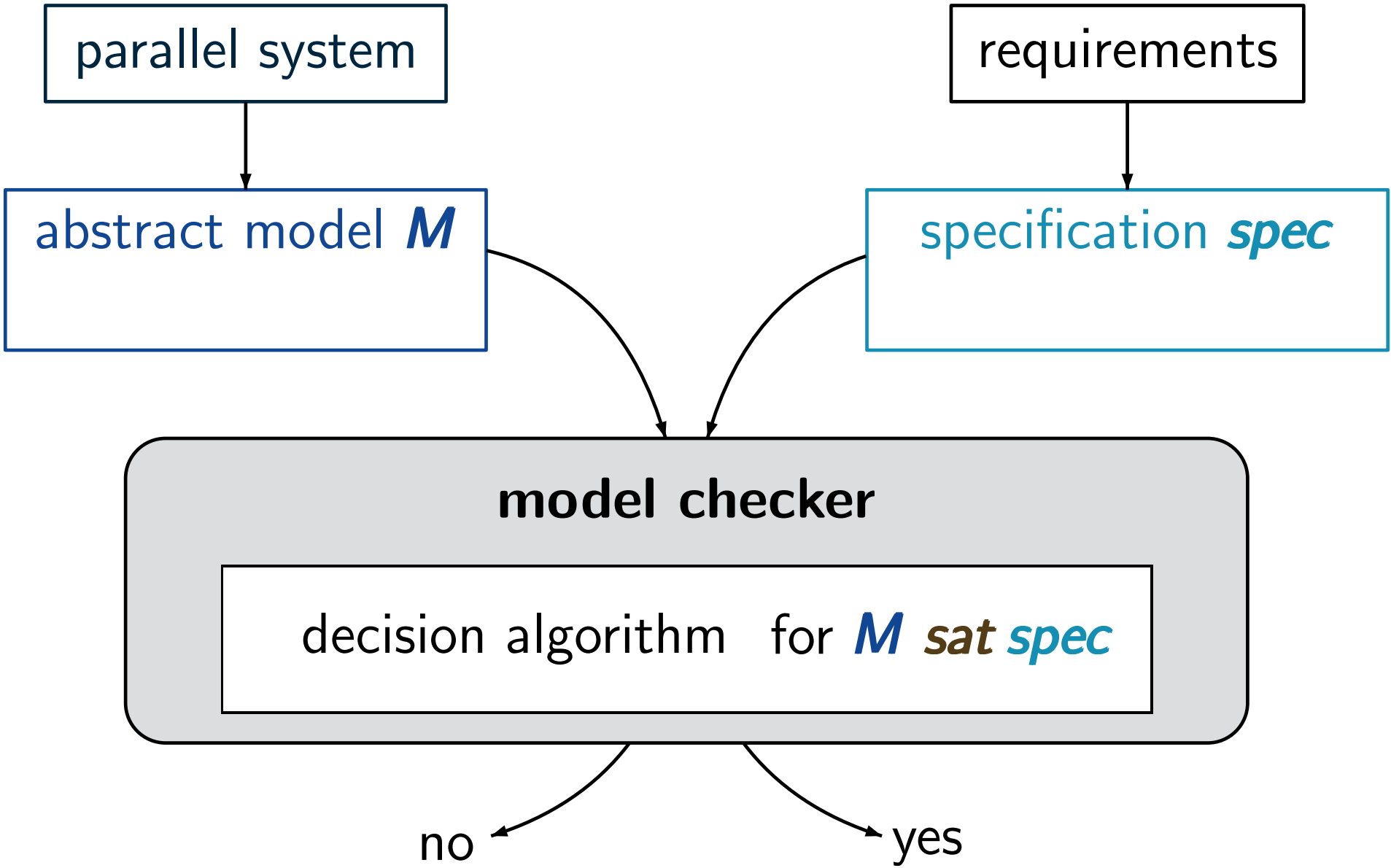
Model checking



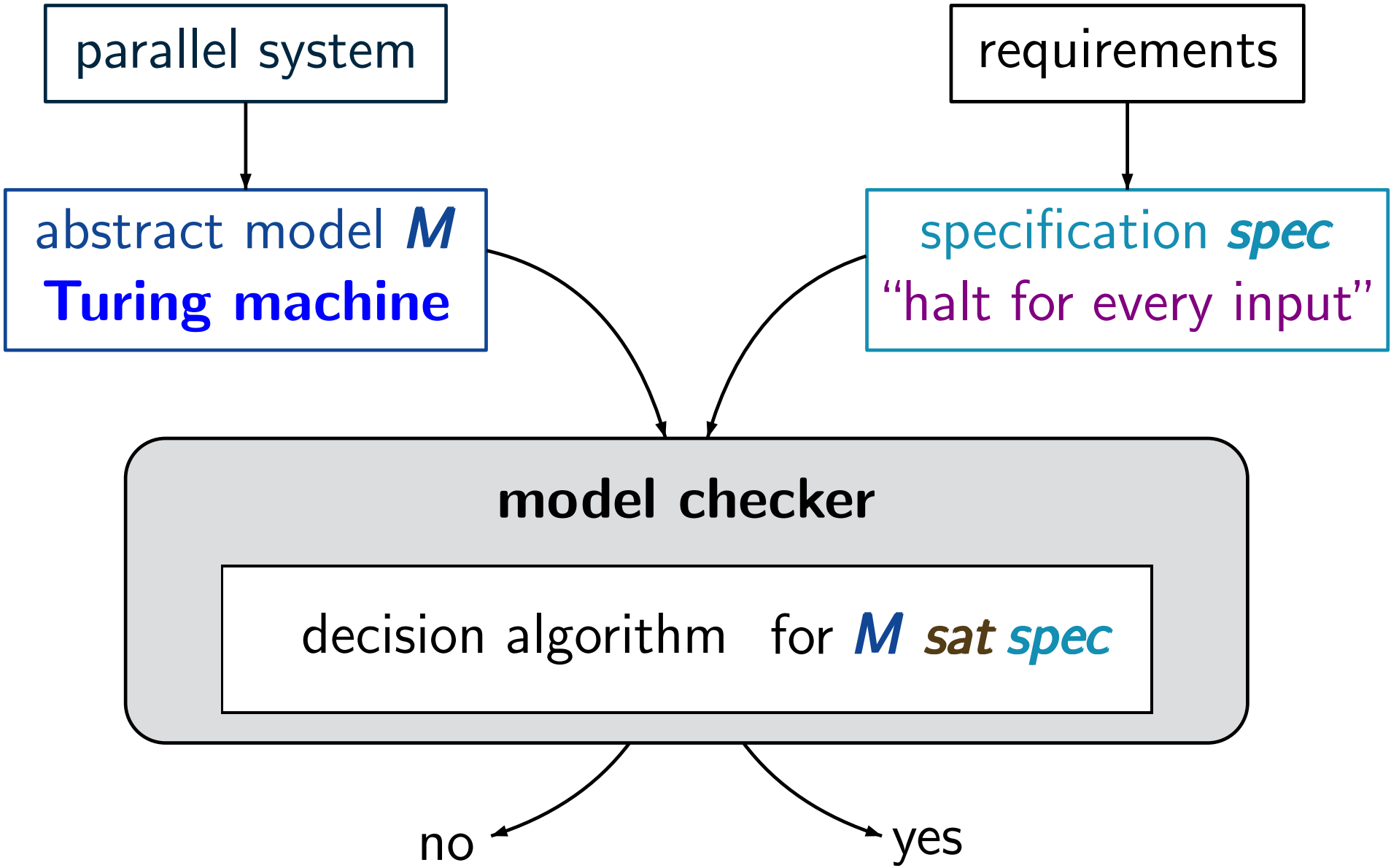
Model checking



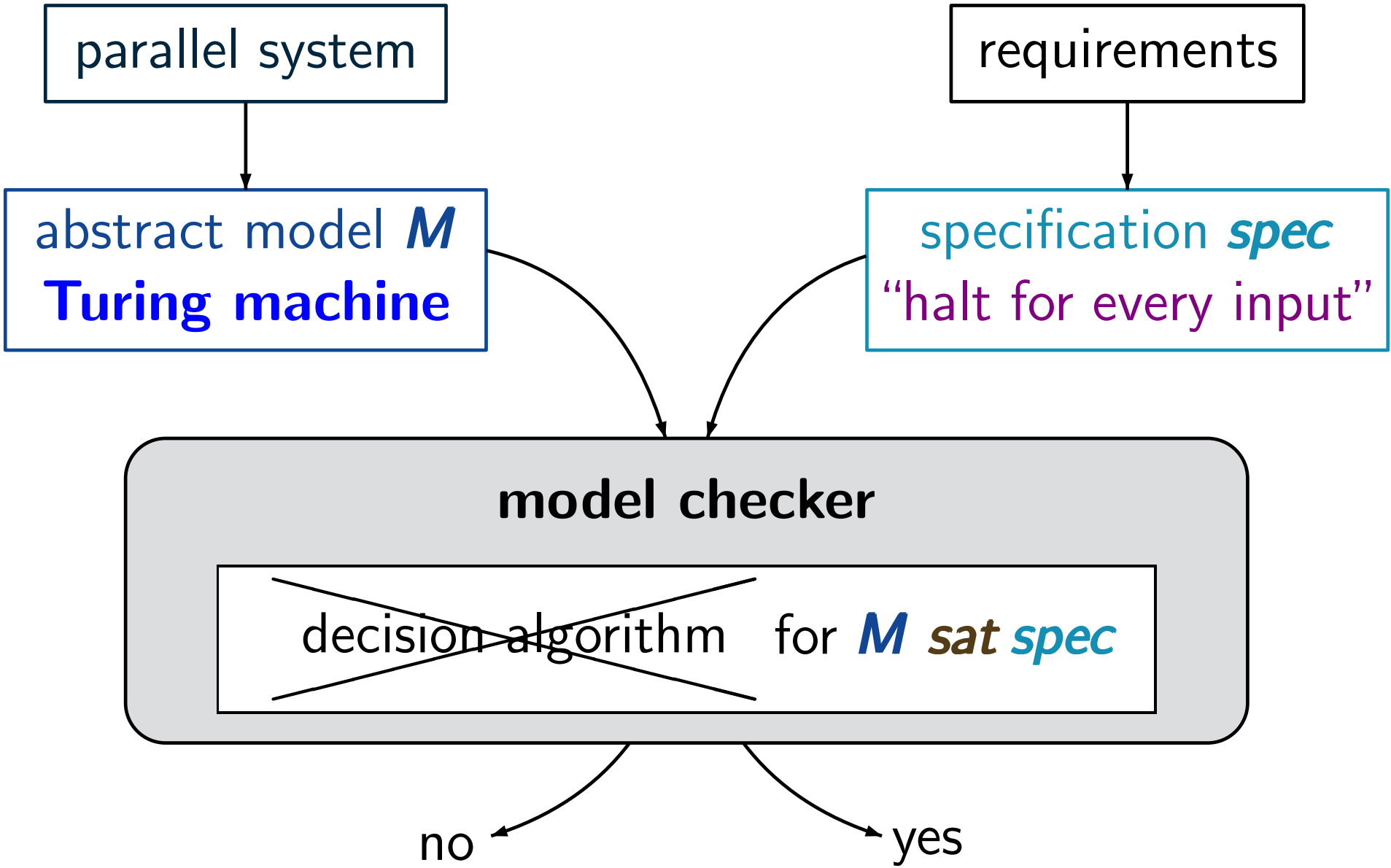
Decidability of the model checking problem?



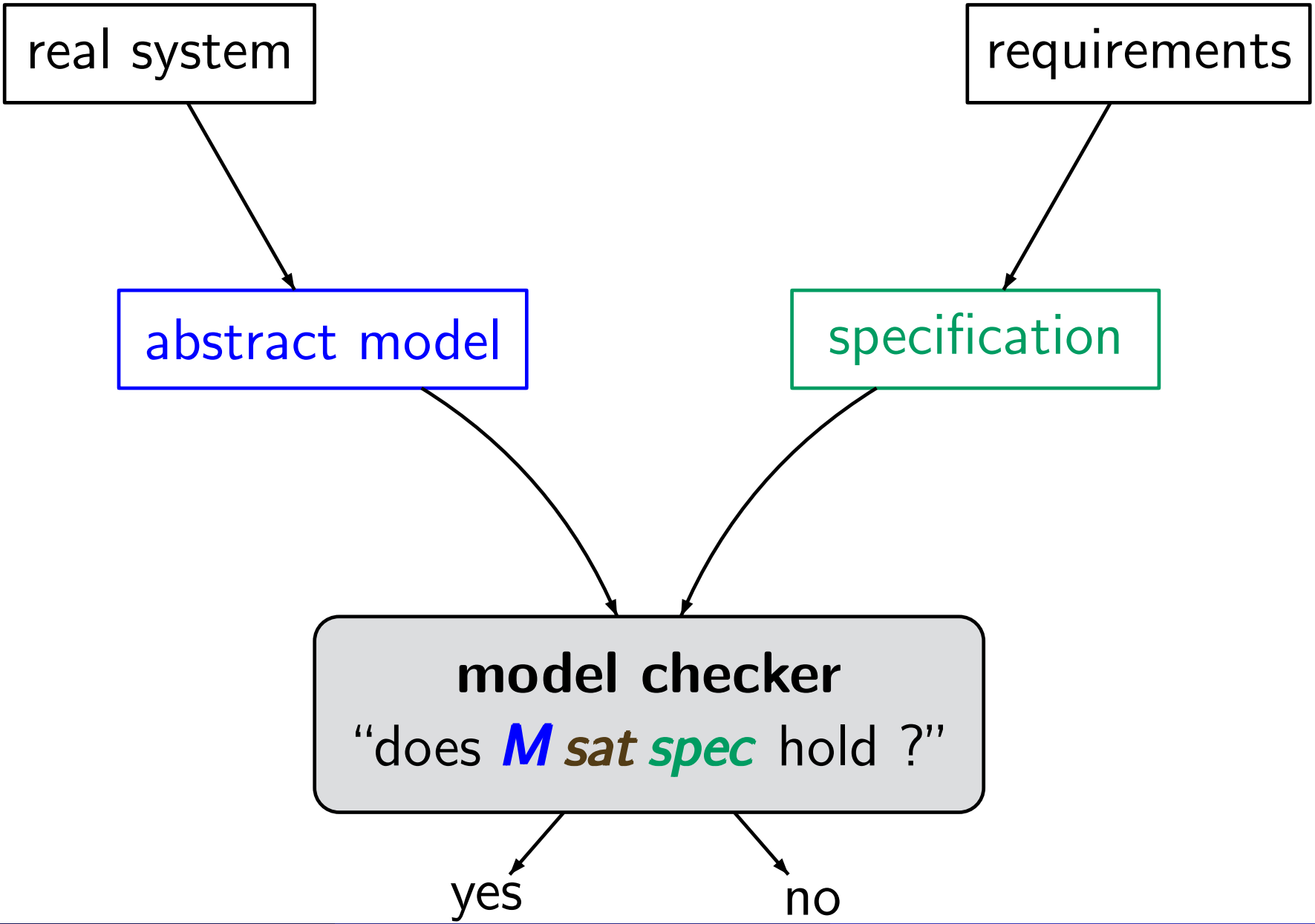
Decidability of the model checking problem?



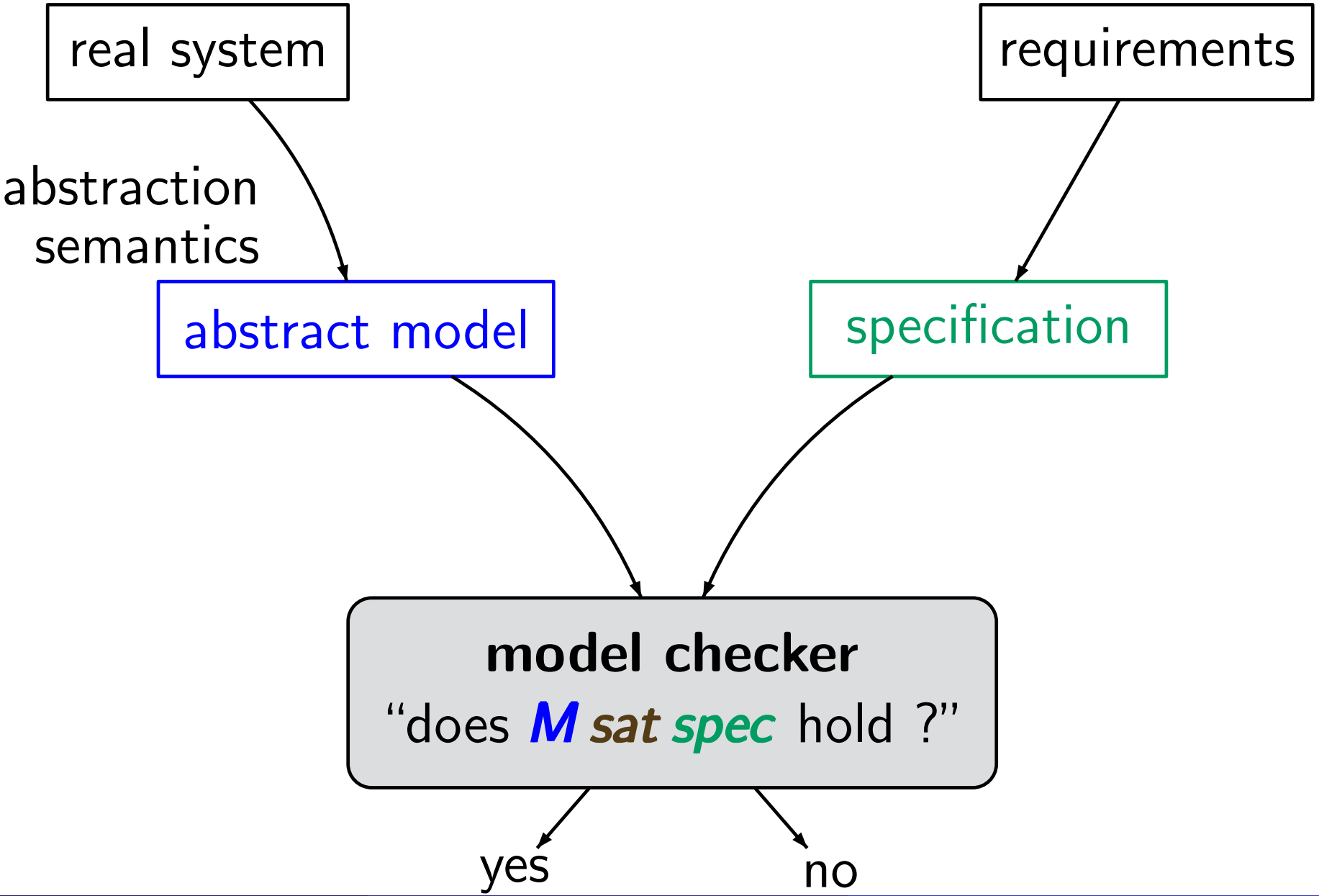
General model checking problem is undecidable



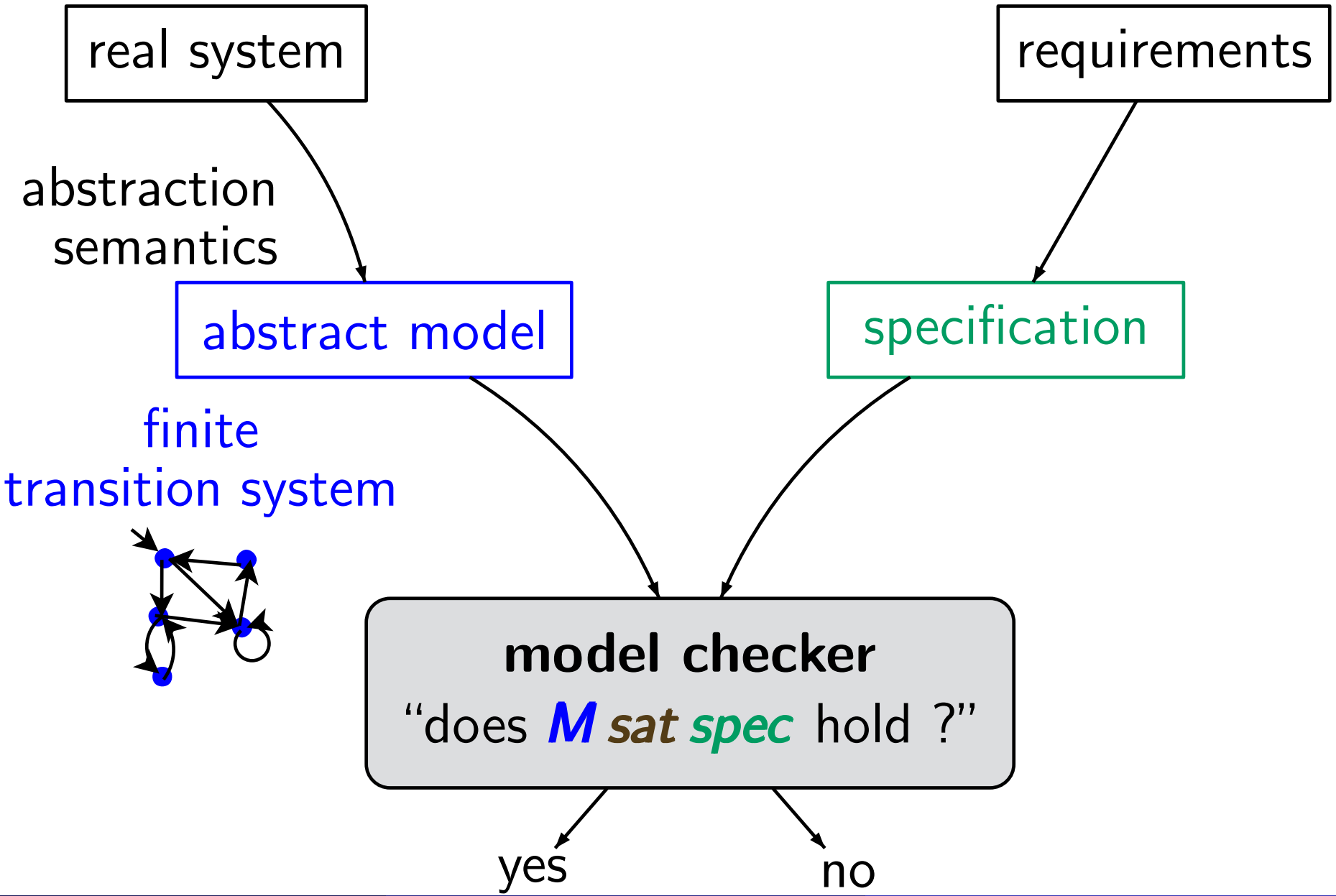
To ensure decidability . . .



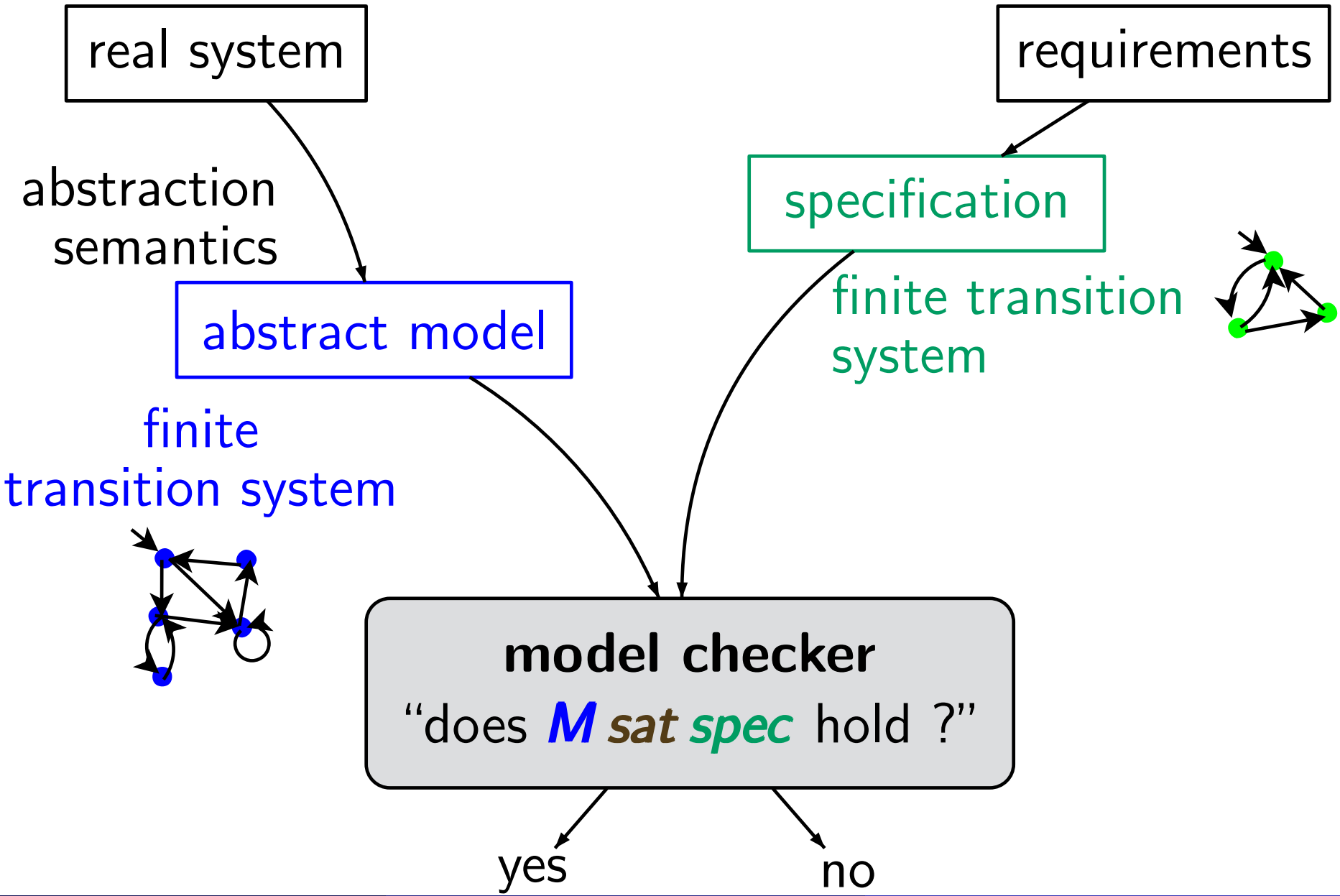
To ensure decidability . . .



To ensure decidability . . .



To ensure decidability . . .



To ensure decidability . . .

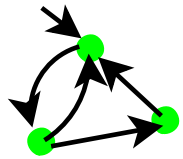
real system

requirements

abstraction
semantics

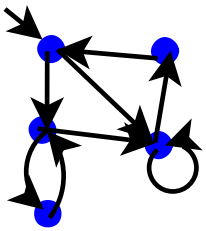
specification

finite transition
system



abstract model

finite
transition system



or temporal formula, e.g.,
$\Box(request \rightarrow \Diamond enter\_crit)$

model checker

"does ***M sat spec*** hold ?"

yes

no

Limiti intrinseci dei metodi formali

- non garantiscono la correttezza del sistema, ma di qualche suo modello astratto (costruito manualmente)
- la formulazione della specifica è eseguita manualmente, può essere incompleta

Verifica automatica

Restrizione teorica: la verifica completamente automatica di tutti i programmi è impossibile.

- restringere la classe di programmi (programmi a stati finiti)
- verificare soltanto le parti cruciali del programma
- semplificare il programma mediante astrazione (manualmente)
- usare metodi di verifica semi-automatica

Sistemi a stati finiti: rappresentabili mediante **sistemi di transizioni** basati su un linguaggio proposizionale.