A Theory Approach to Local-to-Global Algorithms in Spatial Multi-Agent Systems

CS266, Fall 2007
Dan Yamins

Session III: 12.11.2007
Local Checkability

Let $LCR(T)$ denote the minimal radius of a check scheme for it -- this is $T$’s *local check radius*. $T$ is *locally checkable* if $LCR(T)$ is finite.
Local Checkability

Let $\text{LCR}(T)$ denote the minimal radius of a check scheme for it -- this is $T$’s local check radius. $T$ is locally checkable if $\text{LCR}(T)$ is finite.

We saw that a pattern must be locally checkable for there to be a robust solution:

**Proposition.** *If $F$ is a robust solution to 1-dimensional any pattern $T$, then*

$$r(F) \geq \text{LCR}(T).$$
Local Checkability

Let \( \text{LCR}(T) \) denote the minimal radius of a check scheme for it -- this is \( T \)'s local check radius. \( T \) is locally checkable if \( \text{LCR}(T) \) is finite.

We saw that a pattern must be locally checkable for there to be a robust solution:

**Proposition.** *If \( F \) is a robust solution to 1-dimensional any pattern \( T \), then*

\[
    r(F) \geq \text{LCR}(T).
\]

Local checkability is a very general necessary condition for solvability.
Local Checkability

Let $\text{LCR}(T)$ denote the minimal radius of a check scheme for it -- this is $T$’s *local check radius*. $T$ is *locally checkable* if $\text{LCR}(T)$ is finite.

We saw that a pattern must be locally checkable for there to be a robust solution:

**Proposition.** *If $F$ is a robust solution to 1-dimensional any pattern $T$, then*

$$r(F) \geq LCR(T).$$

Local checkability is a very general *necessary condition for solvability*.

The proposition yields both a:
Local Checkability

Let $LCR(T)$ denote the minimal radius of a check scheme for it -- this is $T$’s *local check radius*. $T$ is *locally checkable* if $LCR(T)$ is finite.

We saw that a pattern must be locally checkable for there to be a robust solution:

**Proposition.** *If $F$ is a robust solution to 1-dimensional any pattern $T$, then*

$$r(F) \geq LCR(T).$$

Local checkability is a very general necessary condition for solvability.

The proposition yields both a:

- Sharp existence condition:
Local Checkability

Let $LCR(T)$ denote the minimal radius of a check scheme for it -- this is $T$’s \textit{local check radius}. $T$ is \textit{locally checkable} if $LCR(T)$ is finite.

We saw that a pattern must be locally checkable for there to be a robust solution:

\textbf{Proposition.} \textit{If $F$ is a robust solution to 1-dimensional any pattern $T$, then}

$$r(F) \geq LCR(T).$$

Local checkability is a very general \textbf{necessary condition for solvability}.

The proposition yields both a:

- Sharp existence condition: $LCR(T) = \infty$ means unsolvability
Local Checkability

Let LCR(T) denote the minimal radius of a check scheme for it -- this is T’s *local check radius*. T is *locally checkable* if LCR(T) is finite.

We saw that a pattern must be locally checkable for there to be a robust solution:

**Proposition.** *If F is a robust solution to 1-dimensional any pattern T, then*

\[ r(F) \geq LCR(T). \]

Local checkability is a very general necessary condition for solvability.

The proposition yields both a:

- Sharp existence condition: \( LCR(T) = \infty \) means unsolvability
- and a resource condition:
Local Checkability

Let $LCR(T)$ denote the minimal radius of a check scheme for it -- this is $T$’s *local check radius*. $T$ is *locally checkable* if $LCR(T)$ is finite.

We saw that a pattern must be locally checkable for there to be a robust solution:

**Proposition.** *If $F$ is a robust solution to 1-dimensional any pattern $T$, then*

$$r(F) \geq LCR(T).$$

Local checkability is a very general *necessary condition for solvability.*

The proposition yields both a:

- Sharp existence condition: $LCR(T) = \infty$ means unsolvability
- and a resource condition: $LCR(T)$ is a lower bound.
Local Checkability

Let $LCR(T)$ denote the minimal radius of a check scheme for it -- this is $T$’s *local check radius*. $T$ is *locally checkable* if $LCR(T)$ is finite.

We saw that a pattern must be locally checkable for there to be a robust solution:

**Proposition.** *If $F$ is a robust solution to 1-dimensional any pattern $T$, then

$$r(F) \geq LCR(T).$$

Local checkability is a very general *necessary condition for solvability.*

The proposition yields both a:

- Sharp existence condition: $LCR(T) = \infty$ means unsolvability
- and a resource condition: $LCR(T)$ is a lower bound.

Obvious next questions:
Local Checkability

Let $LCR(T)$ denote the minimal radius of a check scheme for it -- this is T’s local check radius. T is locally checkable if $LCR(T)$ is finite.

We saw that a pattern must be locally checkable for there to be a robust solution:

**Proposition.** *If $F$ is a robust solution to 1-dimensional any pattern $T$, then*

$$r(F) \geq LCR(T).$$

Local checkability is a very general necessary condition for solvability.

The proposition yields both a:

- Sharp existence condition: $LCR(T) = \infty$ means unsolvability
- and a resource condition: $LCR(T)$ is a lower bound.

Obvious next questions: 1) What kinds of patterns are locally checkable?
Local Checkability

Let $LCR(T)$ denote the minimal radius of a check scheme for it -- this is $T$’s local check radius. $T$ is locally checkable if $LCR(T)$ is finite.

We saw that a pattern must be locally checkable for there to be a robust solution:

**Proposition.** *If $F$ is a robust solution to 1-dimensional any pattern $T$, then $
 r(F) \geq LCR(T).$*

Local checkability is a very general necessary condition for solvability.

The proposition yields both a:

- Sharp existence condition: $LCR(T) = \infty$ means unsolvability
- and a resource condition: $LCR(T)$ is a lower bound.

Obvious next questions: 1) What kinds of patterns are locally checkable? And: 2) When is Local Checkability sufficient?
Local Checkability

Q: What kinds of patterns are locally checkable?
Local Checkability

Q: What kinds of patterns are locally checkable?
A: 1-D: Basically, various combinations of repeat patterns.

Combining repeat, proportionate, fractal, and ellipsoid patterns, we have a “locally checkable vector graphics language” in higher-D lattices.
Local Checkability

Q: What kinds of patterns are locally checkable?
A: 1-D: Basically, various combinations of repeat patterns.
    Higher -D: lots of stuff.

Combining repeat, proportionate, fractal, and ellipsoid patterns, we have a "locally checkable vector graphics language" in higher-D lattices.
Local Checkability

Local checkability is the first step of a “global-to-local” compiler:
Local Checkability

Local checkability is the first step of a “global-to-local” compiler:

Input: Pattern
Local Checkability

Local checkability is the first step of a “global-to-local” compiler:

Input: Pattern
Local Checkability

Local checkability is the first step of a “global-to-local” compiler:

Input: Pattern

Output:
Local Checkability

Local checkability is the first step of a “global-to-local” compiler:

**Input: Pattern**

![Pattern Examples]

**Output:**

- min radius = 2 (3 states)
- min radius = 1 (2 states)
- min radius = 2 (2 states)
- min radius = 4 (3 states)
Local Checkability

Local checkability is the first step of a “global-to-local” compiler:

Input: Pattern

Output:

- min radius = 2 (3 states) \( \Theta_{Stripe} \)
- min radius = 1 (2 states) \( \Theta_{Watermelon} \)
- min radius = 2 (2 states) \( \Theta_{Cross} \)
- min radius = 4 (3 states) \( \Theta_{DV-Split} \)
Local Checkability

Local checkability is the first step of a “global-to-local” compiler:

Input: Pattern

Output:

- \( \Theta_{\text{Stripe}} \) with min radius = 2 (3 states)
- \( \Theta_{\text{Watermelon}} \) with min radius = 1 (2 states)
- \( \Theta_{\text{Cross}} \) with min radius = 2 (2 states)
- \( \Theta_{\text{DV-Split}} \) with min radius = 4 (3 states)

Next step: actually constructing local rule solutions.
Local Rule Constructions

Let’s first restrict attention to the 1-D directed lines:
Local Rule Constructions

Let’s first restrict attention to the 1-D directed lines:
Local Rule Constructions

Let’s first restrict attention to the 1-D directed lines:

We want to show that:
Local Rule Constructions

Let’s first restrict attention to the 1-D directed lines:

We want to show that:

Every one-dimensional locally checkable pattern is robustly solvable.
Local Rule Constructions

Let’s first restrict attention to the 1-D directed lines:

We want to show that:

Every one-dimensional locally checkable pattern is robustly solvable.

For each check scheme $\Theta$ we will find a local rule $F_\Theta$ that is a robust solution to the pattern generated by $\Theta$. 
Local Rule Constructions

Let’s first restrict attention to the 1-D directed lines:

We want to show that:

Every one-dimensional locally checkable pattern is robustly solvable.

For each check scheme $\Theta$ we will find a local rule $F_{\Theta}$ that is a robust solution to the pattern generated by $\Theta$.

Let’s start with the repeat patterns:
Local Rule Constructions

Let’s first restrict attention to the 1-D directed lines:

We want to show that:

Every one-dimensional locally checkable pattern is robustly solvable.

For each check scheme $Θ$ we will find a local rule $F_Θ$ that is a robust solution to the pattern generated by $Θ$.

Let’s start with the repeat patterns:

$$T_q = \{q, q \circ q, \ldots, q^n, \ldots\}$$
Local Rule Constructions

Let’s first restrict attention to the 1-D directed lines:

We want to show that:

Every one-dimensional locally checkable pattern is robustly solvable.

For each check scheme $\Theta$ we will find a local rule $F_\Theta$ that is a robust solution to the pattern generated by $\Theta$.

Let’s start with the repeat patterns:

$$T_q = \{q, q \circ q, \ldots, q^n, \ldots\}$$

where $q$ is a finite “unit”.

Thursday, November 28, 13
Local Rule Constructions
Local Rule Constructions

Given a radius-R local check scheme and a local ball B, define:
Given a radius-R local check scheme and a local ball $B$, define:

$$
\nabla_\Theta(B)^+ = \begin{cases} 
i, & \text{if } B \circ i \text{ is consistent with } \Theta \\ 
B(2R), & \text{otherwise} 
\end{cases}
$$
Local Rule Constructions

Given a radius-$R$ local check scheme and a local ball $B$, define:

$$\nabla_{\Theta}(B)^+ = \begin{cases} i, & \text{if } B \circ i \text{ is consistent with } \Theta \\ B(2R), & \text{otherwise} \end{cases}$$

Repeat patterns have a well-defined gradient at every point.
Local Rule Constructions

Given a radius-R local check scheme and a local ball B, define:

$$\nabla_\Theta(B)^+ = \begin{cases} 
  i, & \text{if } B \circ i \text{ is consistent with } \Theta \\
  B(2R), & \text{otherwise}
\end{cases}$$

Repeat patterns have a well-defined gradient at every point.

For example: 1000-repeat pattern (which has an R=2 check scheme),
Local Rule Constructions

Given a radius-$R$ local check scheme and a local ball $B$, define:

$$\nabla_{\Theta}(B)^+ = \begin{cases} 
i, & \text{if } B \circ i \text{ is consistent with } \Theta \\ B(2R), & \text{otherwise} \end{cases}$$

Repeat patterns have a well-defined gradient at every point.

For example: 1000-repeat pattern (which has an $R=2$ check scheme),

$$\nabla_{\Theta}(100)^+ = 0 \quad \nabla_{\Theta}(001)^+ = 0 \quad \text{Otherwise,}$$

$$\nabla_{\Theta}(000)^+ = 1 \quad \nabla_{\Theta}(010)^+ = 0 \quad \nabla_{\Theta}(b) = b(3)$$
Local Rule Constructions
Local Rule Constructions

Now for a radius 2R ball $B$, simply define:
Local Rule Constructions

Now for a radius 2R ball $B$, simply define:

$$F(B) = \nabla^+_{\Theta}(B(1 : 2r))$$
Local Rule Constructions

Now for a radius $2R$ ball $B$, simply define:

$$F(B) = \nabla^+_{\Theta}(B(1 : 2r))$$

[Diagram showing a ball $2r(\Theta)$ with points marked in blue and red]
Now for a radius $2R$ ball $B$, simply define:

$$F(B) = \nabla^+_{\Theta}(B(1 : 2r))$$
Now for a radius $2R$ ball $B$, simply define:

$$F(B) = \nabla^+_{\Theta}(B(1:2r))$$
Now for a radius 2R ball B, simply define:

$$F(B) = \nabla^+_{\Theta}(B(1:2r))$$
Local Rule Constructions

Now for a radius 2R ball $B$, simply define:

$$F(B) = \nabla_{\Theta}^+(B(1:2r))$$
Local Rule Constructions

Now for a radius 2R ball $B$, simply define:

$$F(B) = \nabla^+_{\Theta}(B(1 : 2r))$$
Now for a radius $2R$ ball $B$, simply define:

$$F(B) = \nabla_{\Theta}^{+} (B(1:2r))$$
Now for a radius $2R$ ball $B$, simply define:

$$F(B) = \nabla^+_\Theta (B(1:2r))$$
Local Rule Constructions

Now for a radius $2R$ ball $B$, simply define:

$$F(B) = \nabla^+_\Theta (B(1:2r))$$

Gradient waves ...
Local Rule Constructions

Now for a radius 2R ball B, simply define:

\[ F(B) = \nabla^+_{\Theta}(B(1:2r)) \]

Gradient waves ... Multiple “waves” all at once in actuality ....
Local Rule Constructions

Now for a radius 2R ball B, simply define:

$$F(B) = \nabla^+_\Theta (B(1 : 2r))$$

Gradient waves ... Multiple “waves” all at once in actuality ....

**Proposition.** Any 1-dimensional repeat pattern $T_q$ has a radius-2$|q|$ gradient-based robust solution.
Local Rule Constructions

Now, suppose we were given the pattern
Local Rule Constructions

Now, suppose we were given the pattern

$T_{100} \cup T_{1000} = \{(100)^n\} \cup \{(1000)^n\}$

the ‘OR’ of two repeat patterns.
Local Rule Constructions

Now, suppose we were given the pattern

\[ T_{100} \cup T_{1000} = \{(100)^n\} \cup \{(1000)^n\} \]

the ‘OR’ of two repeat patterns.

Thursday, November 28, 13
Local Rule Constructions

Now, suppose we were given the pattern

\[ T_{100} \cup T_{1000} = \{(100)^n\} \cup \{(1000)^n\} \]

the ‘OR’ of two repeat patterns.

What does “robustly solving” this pattern really mean?
Local Rule Constructions

Now, suppose we were given the pattern

\[ T_{100} \cup T_{1000} = \{(100)^n\} \cup \{(1000)^n\} \]

the ‘OR’ of two repeat patterns.

What does “robustly solving” this pattern really mean?

It means: 1) figuring out whether the system is a multiple of 3 or 4 in size (if either) and 2) constructing the correct pattern.
Local Rule Constructions

Now, suppose we were given the pattern

\[ T_{100} \cup T_{1000} = \{(100)^n\} \cup \{(1000)^n\} \]

the ‘OR’ of two repeat patterns.

What does “robustly solving” this pattern really mean?

It means: 1) figuring out whether the system is a multiple of 3 or 4 in size (if either) and 2) constructing the correct pattern.

For some systems (sizes multiple of 12), both subpatterns can work.
Local Rule Constructions

In analogy with the definition $\nabla^+_\Theta$ from before, let:

$$\nabla^-\Theta(B) = \begin{cases} i, & \text{if } i \circ B \text{ is consistent with } \Theta \\ B(1), & \text{otherwise} \end{cases}$$
Local Rule Constructions

In analogy with the definition $\nabla_\Theta^+$ from before, let:

$$\nabla_\Theta(B)^- = \begin{cases} i, & \text{if } i \circ B \text{ is consistent with } \Theta \\ B(1), & \text{otherwise} \end{cases}$$

Now, construct a local rule $F$ which:
Local Rule Constructions

In analogy with the definition $\nabla^+_{\Theta}$ from before, let:

$$\nabla(\Theta)(B)^- = \begin{cases} i, & \text{if } i \circ B \text{ is consistent with } \Theta \\ B(1), & \text{otherwise} \end{cases}$$

Now, construct a local rule $F$ which:

- Generates a $\nabla^+_{T_{100}}$-wave from the left:
Local Rule Constructions

In analogy with the definition $\nabla_\Theta^+$ from before, let:

$$\nabla_\Theta(B)^- = \begin{cases} i, & \text{if } i \circ B \text{ is consistent with } \Theta \\ B(1), & \text{otherwise} \end{cases}$$

Now, construct a local rule $F$ which:

- Generates a $\nabla_{T_{100}}^+$-wave from the left:

- Generates a $\nabla_{T_{1000}}^-$-wave from the right:
Local Rule Constructions

In analogy with the definition $\nabla_\Theta^+$ from before, let:

$$\nabla_\Theta(B)^- = \begin{cases} i, & \text{if } i \circ B \text{ is consistent with } \Theta \\ B(1), & \text{otherwise} \end{cases}$$

Now, construct a local rule $F$ which:

- Generates a $\nabla_{T_{100}}^+$-wave from the left:

- Generates a $\nabla_{T_{1000}}^-$-wave from the right:

- Has boundary reflection rules:
  $$\nabla_{T_{100}}^+ \rightarrow \nabla_{T_{1000}}^- \text{ at the right-end, if not correct}$$
Local Rule Constructions

In analogy with the definition $\nabla^+_\Theta$ from before, let:

$$\nabla^+_\Theta(B) = \begin{cases} i, & \text{if } i \circ B \text{ is consistent with } \Theta \\ B(1), & \text{otherwise} \end{cases}$$

Now, construct a local rule $F$ which:

- Generates a $\nabla^+_{T_{100}}$-wave from the left:
  $$\nabla^+_{100}$$

- Generates a $\nabla^-_{T_{1000}}$-wave from the right:
  $$\nabla^-_{1000}$$

- Has boundary reflection rules:
  $$\nabla^+_{T_{100}} \rightarrow \nabla^-_{T_{1000}} \quad \text{at the right-end, if not correct}$$
  $$\nabla^-_{T_{1000}} \rightarrow \nabla^+_{T_{100}} \quad \text{at the left-end, if not correct}$$
Local Rule Constructions

In analogy with the definition $\nabla_{\Theta}^+$ from before, let:

$$\nabla_{\Theta}(B)^{-} = \begin{cases} i, & \text{if } i \circ B \text{ is consistent with } \Theta \\ B(1), & \text{otherwise} \end{cases}$$

Now, construct a local rule $F$ which:

- Generates a $\nabla_{T_{100}}^+$-wave from the left:

  $F = \nabla_{100}^+ + \nabla_{1000}^-$

- Generates a $\nabla_{T_{1000}}^-$-wave from the right:

- Has boundary reflection rules:

  $\nabla_{T_{100}}^+ \rightarrow \nabla_{T_{1000}}^-$ at the right-end, if not correct

  $\nabla_{T_{1000}}^- \rightarrow \nabla_{T_{100}}^+$ at the left-end, if not correct
Consider the pattern

\[ T_{100} \cdot T_{1000} = \{(100)^n (1000)^m\} \]
Local Rule Constructions

Consider the pattern

\[ T_{100} \cdot T_{1000} = \{(100)^n(1000)^m\} \]
Local Rule Constructions

Consider the pattern

$$T_{100} \cdot T_{1000} = \{(100)^n (1000)^m\}$$

What does “robustly solving” this pattern really mean?
Local Rule Constructions

Consider the pattern

\[ T_{100} \cdot T_{1000} = \{(100)^n (1000)^m\} \]

What does “robustly solving” this pattern really mean?
Local Rule Constructions

Consider the pattern

\[ T_{100} \cdot T_{1000} = \{(100)^n (1000)^m\} \]

What does “robustly solving” this pattern really mean?
Local Rule Constructions

Consider the pattern

\[ T_{100} \cdot T_{1000} = \{(100)^n (1000)^m\} \]

What does “robustly solving” this pattern really mean?

Choosing between globally different but locally similar alternatives,
Local Rule Constructions

Consider the pattern

\[ T_{100} \cdot T_{1000} = \{(100)^n(1000)^m\} \]

What does “robustly solving” this pattern really mean?

Choosing between globally different but locally similar alternatives, as well making the actual construction.
Local Rule Constructions

Consider the pattern

\[ T_{100} \cdot T_{1000} = \{(100)^n(1000)^m\} \]

What does “robustly solving” this pattern really mean?

Choosing between globally different but locally similar alternatives, as well making the actual construction.

Unlike single-choice patterns, not every 2-ball has a unique successor.
Local Rule Constructions

Consider the pattern

$$T_{100} \cdot T_{1000} = \{(100)^n (1000)^m\}$$

What does “robustly solving” this pattern really mean?

Choosing between globally different but locally similar alternatives, as well making the actual construction.

Unlike single-choice patterns, not every 2-ball has a unique successor.
Local Rule Constructions

Consider the pattern

$$T_{100} \cdot T_{1000} = \{(100)^n (1000)^m\}$$

What does “robustly solving” this pattern really mean?

Choosing between globally different but locally similar alternatives, as well as making the actual construction.

Unlike single-choice patterns, not every 2-ball has a unique successor.

\[\n+\quad \text{isn’t always well-defined.} \]
Local Rule Constructions

Q: What would happen if we tried to generalize the gradient wave approach?
Local Rule Constructions

Q: What would happen if we tried to generalize the gradient wave approach?

-
Local Rule Constructions

Q: What would happen if we tried to generalize the gradient wave approach?

• Have to choose a value for $\nabla^+_\Theta$, say 0.
Local Rule Constructions

Q: What would happen if we tried to generalize the gradient wave approach?

• Have to choose a value for $\nabla^+ \Theta$, say 0.

•
Local Rule Constructions

Q: What would happen if we tried to generalize the gradient wave approach?

• Have to choose a value for $\nabla^+_\Theta$, say 0.

• Number of 100 repeats toward left end has to change.
Local Rule Constructions

Q: What would happen if we tried to generalize the gradient wave approach?

- Have to choose a value for $\nabla^+_\Theta$, say 0.
- Number of 100 repeats toward left end has to change.
Q: What would happen if we tried to generalize the gradient wave approach?

- Have to choose a value for $\nabla^+_\Theta$, say 0.
- Number of 100 repeats toward left end has to change.
Local Rule Constructions

Q: What would happen if we tried to generalize the gradient wave approach?

• Have to choose a value for $\nabla_{\Theta}^+$, say 0.
• Number of 100 repeats toward left end has to change.
Local Rule Constructions

Q: What would happen if we tried to generalize the gradient wave approach?

- Have to choose a value for $\nabla^+_\theta$, say 0.
- Number of 100 repeats toward left end has to change.

Thursday, November 28, 13
Q: What would happen if we tried to generalize the gradient wave approach?

- Have to choose a value for $\nabla^+_\Theta$, say 0.
- Number of 100 repeats toward left end has to change.
**Local Rule Constructions**

Q: What would happen if we tried to generalize the gradient wave approach?

- Have to choose a value for $\nabla^+_\Theta$, say 0.
- Number of 100 repeats toward left end has to change.

Actually, this story *can* be captured by local rules ...
Local Rule Constructions
Local Rule Constructions

With 10 local (partial) rules we can implement “a Naive Backtracking search with a self-organized virtual distributed Turing machine.”
Local Rule Constructions

With 10 local (partial) rules we can implement “a Naive Backtracking search with a self-organized virtual distributed Turing machine.”

• Rule 1: Head birth at local error
Local Rule Constructions

With 10 local (partial) rules we can implement “a Naive Backtracking search with a self-organized virtual distributed Turing machine.”

- **Rule 1**: Head birth at local error

- **Rules 2-3**: Head propagation
Local Rule Constructions

With 10 local (partial) rules we can implement “a Naive Backtracking search with a self-organized virtual distributed Turing machine.”

- **Rule 1:** Head birth at local error

- **Rules 2-3:** Head propagation

- **Rule 4:** Head halting
Local Rule Constructions

With 10 local (partial) rules we can implement “a Naive Backtracking search with a self-organized virtual distributed Turing machine.”

- **Rule 1**: Head birth at local error
- **Rules 2-3**: Head propagation
- **Rule 4**: Head halting
- **Rule 5**: Reversal
Local Rule Constructions

With 10 local (partial) rules we can implement “a Naive Backtracking search with a self-organized virtual distributed Turing machine.”

- **Rule 1:** Head birth at local error
- **Rules 2-3:** Head propagation
- **Rule 4:** Head halting
- **Rule 5:** Reversal
- **Rule 6:** Left propagation
With 10 local (partial) rules we can implement “a Naive Backtracking search with a self-organized virtual distributed Turing machine.”

- **Rule 1**: Head birth at local error
- **Rules 2-3**: Head propagation
- **Rule 4**: Head halting
- **Rule 5**: Reversal
- **Rule 6**: Left propagation
- **Rules 7-9**: “Upclick” Re-reversal

Thursday, November 28, 13
Local Rule Constructions

With 10 local (partial) rules we can implement “a Naive Backtracking search with a self-organized virtual distributed Turing machine.”

- Rule 1: Head birth at local error
- Rules 2-3: Head propagation
- Rule 4: Head halting
- Rule 5: Reversal
- Rule 6: Left propagation
- Rules 7-9: “Upclick” Re-reversal
- Rule 10: Reset
Rule 1: If
- $B(-2r - 1 : -1)$ satisfies $\Theta$, and
- $B(-2r : 0)$ does NOT satisfy $\Theta$,
then $F_{\Theta}(B) = \triangleright$.

Rule 2: If
- $B(0) = B(1) = \triangleright$, and
- $B(2r - 1 : -1)$ satisfies $\Theta$,
then $F_{\Theta}(b) = \nabla_{\Theta}(b)^+$ when the latter exists.

Rule 3: If
- $B(-1) = \triangleright$ and
- $B(-2r - 2 : -2)$ satisfies $\Theta$,
then $F_{\Theta}(B) = \triangleright$.

Rule 4: For the right-end agent, if
- $B(0) = \triangleright$, and,
- $B(-2r - 1 : -1)$ satisfies $\Theta$,
then $F_{\Theta} = \eta(B)$ when the latter exists.

Rule 5: If
- as in Rule 2, BUT $\nabla_{\Theta}(B_{-})^+$ does not exist, or
- as in Rule 4, BUT $\eta[B]$ does not exist,
then $F_{\Theta}(B) = \triangleleft$.

Rule 6: If
- $B(1) = \triangleleft$, and
- $B(-2r - 1 : -1)$ and $B(-2r : 0)$ both satisfy $\Theta$, and
- $B(0) = M(B)$,
then $F_{\Theta}(B) = \triangleleft$.

Rule 7: If
- $B(0) = \triangleleft$, and
- $B(-2r - 2 : -2)$ and $B(-2r - 1 : -1)$ both satisfy $\Theta$, and
- $B(-1) \neq M(B(-2r - 1 : -1))$,
then $F_{\Theta}(B) = \Delta_{B(-1)}$.

Rule 8: If
- $B(1) = \Delta_{B(0)}$ and
- $B(-2r : 0)$ satisfies $\Theta$, and
- $B(0) \neq M(B(-2r : 0))$,
then $F_{\Theta}(B) = \Theta(B(-2r : 0))^+$.

Rule 9: If
- $B(0) = \Delta_j$ for some $j \neq B(-1)$, or
- $B(0) = \Delta_{B(-1)}$ and $B(-1) = M(B(-2r - 1 : -1))$,
then $F(B) = \triangleright$.

Rule 10: For the agent is the left-end, if $B(0) = \triangleleft$ then $F(B) = \triangleright$. 

Thursday, November 28, 13
Local Rule Constructions
Local Rule Constructions

I call the rule $F_\Theta$ thereby defined the “naive backtracking rule.”
Local Rule Constructions

I call the rule $F_\Theta$ thereby defined the “naive backtracking rule.”

**Theorem.** Local checkability is a necessary and sufficient condition for robust solvability in one dimension; and any solvable pattern $T$ has a solution $F$ with

$$r(F) \leq 4LCR(T) + 8.$$
Local Rule Constructions

I call the rule $F_\Theta$ thereby defined the “naive backtracking rule.”

**Theorem.** Local checkability is a necessary and sufficient condition for robust solvability in one dimension; and any solvable pattern $T$ has a solution $F$ with

$$r(F) \leq 4LCR(T) + 8.$$ 

It is robust to all initial condition and timing perturbations.
Local Rule Constructions

I call the rule $F_\Theta$ thereby defined the “naive backtracking rule.”

**Theorem.** Local checkability is a necessary and sufficient condition for robust solvability in one dimension; and any solvable pattern $T$ has a solution $F$ with

$$r(F) \leq 4LCR(T) + 8.$$

It is robust to all initial condition and timing perturbations.

Architecturally:
Local Rule Constructions

I call the rule $F_\Theta$ thereby defined the “naive backtracking rule.”

Theorem. Local checkability is a necessary and sufficient condition for robust solvability in one dimension; and any solvable pattern $T$ has a solution $F$ with

$$r(F) \leq 4LCR(T) + 8.$$ 

It is robust to all initial condition and timing perturbations.

Architecturally:

- Simple check schemes have simple gradient solutions. (Fast)
Local Rule Constructions

I call the rule $F_{\Theta}$ thereby defined the “naive backtracking rule.”

**Theorem.** Local checkability is a necessary and sufficient condition for robust solvability in one dimension; and any solvable pattern $T$ has a solution $F$ with

$$r(F) \leq 4LCR(T) + 8.$$

It is robust to all initial condition and timing perturbations.

Architecturally:

- Simple check schemes have simple gradient solutions. (Fast)
- More complicated check scheme have multi-gradient solutions. (Fast)
Local Rule Constructions

I call the rule $F_0$ thereby defined the “naive backtracking rule.”

**Theorem.** Local checkability is a necessary and sufficient condition for robust solvability in one dimension; and any solvable pattern $T$ has a solution $F$ with

$$r(F) \leq 4LCR(T) + 8.$$  

It is robust to all initial condition and timing perturbations.

**Architecturally:**

- Simple check schemes have simple gradient solutions. (Fast)
- More complicated check scheme have multi-gradient solutions. (Fast)
- The most complicated check schemes can be solved by a virtual and distributed Turing machine -- with *multiple* heads moving around the system concurrently. (Can be slow)
Local Rule Constructions

I call the rule $F_{\Theta}$ thereby defined the “naive backtracking rule.”

**Theorem.** Local checkability is a necessary and sufficient condition for robust solvability in one dimension; and any solvable pattern $T$ has a solution $F$ with

$$r(F) \leq 4\text{LCR}(T) + 8.$$ 

It is robust to all initial condition and timing perturbations.

**Architecturally:**

- Simple check schemes have simple gradient solutions. (Fast)
- More complicated check scheme have multi-gradient solutions. (Fast)
- The most complicated check schemes can be solved by a virtual and distributed Turing machine -- with *multiple* heads moving around the system concurrently. (Can be slow)

**Proposition.** Every 1-dimensional check scheme $\Theta$ has a “smart” solution $\hat{F}_{\Theta}$ solving $\Theta$ with worst-case runtime $O(n)$. 
Local Rule Constructions

I call the rule $F_{\Theta}$ thereby defined the “naive backtracking rule.”

**Theorem.** Local checkability is a necessary and sufficient condition for robust solvability in one dimension; and any solvable pattern $T$ has a solution $F$ with

$$r(F) \leq 4LCR(T) + 8.$$ 

It is robust to all initial condition and timing perturbations.

Architecturally:

- Simple check schemes have simple gradient solutions. (Fast)
- More complicated check scheme have multi-gradient solutions. (Fast)
- The most complicated check schemes can be solved by a virtual and distributed Turing machine -- with *multiple* heads moving around the system concurrently. (Can be slow)

**Proposition.** Every 1-dimensional check scheme $\Theta$ has a “smart” solution $\hat{F}_{\Theta}$ solving $\Theta$ with worst-case runtime $O(n)$.

In the smart rule, virtual heads “glean information” as they move through the space, make smart backtracking interactions, and need only finitely many “sweeps.”
Local Rule Constructions

Q: How do these results hold up in other spaces?
Local Rule Constructions

Q: How do these results hold up in other spaces? 2-D lattices.
Local Rule Constructions

Q: How do these results hold up in other spaces? 2-D lattices.

Higher-D repeat patterns still have well-defined gradients in every direction:
Local Rule Constructions

Q: How do these results hold up in other spaces? 2-D lattices.

Higher-D repeat patterns still have well-defined gradients in every direction:
Local Rule Constructions

Q: How do these results hold up in other spaces? 2-D lattices.

Higher-D repeat patterns still have well-defined gradients in every direction:
Local Rule Constructions

Q: How do these results hold up in other spaces? 2-D lattices.

Higher-D repeat patterns still have well-defined gradients in every direction:
Q: How do these results hold up in other spaces? **2-D lattices.**

Higher-D repeat patterns **still** have well-defined gradients in every direction:
Local Rule Constructions

Q: How do these results hold up in other spaces? 2-D lattices.

Higher-D repeat patterns still have well-defined gradients in every direction:

- Simply pick a basis set of directions, say $v_1$ and $v_2$ and define
Local Rule Constructions

Q: How do these results hold up in other spaces? 2-D lattices.

Higher-D repeat patterns still have well-defined gradients in every direction:

Simply pick a basis set of directions, say $\vec{v}_1$ and $\vec{v}_2$ and define

$$F_\Theta(B) = \nabla_{\vec{v}_1}(B) + \nabla_{\vec{v}_2}(B)$$
Local Rule Constructions

Q: How do these results hold up in other spaces? 2-D lattices.

Higher-D repeat patterns still have well-defined gradients in every direction:

Simply pick a basis set of directions, say \( \vec{v}_1 \) and \( \vec{v}_2 \) and define

\[
F_{\Theta}(B) = \nabla_{\Theta}^{\vec{v}_1}(B) + \nabla_{\Theta}^{\vec{v}_2}(B)
\]
Local Rule Constructions

Q: How do these results hold up in other spaces? **2-D lattices.**

Higher-D repeat patterns **still** have well-defined gradients in every direction:

Simply pick a *basis* set of directions, say \( \vec{v}_1 \) and \( \vec{v}_2 \) and define

\[
F_{\Theta}(B) = \nabla_{\Theta}^{\vec{v}_1}(B) + \nabla_{\Theta}^{\vec{v}_2}(B)
\]
Local Rule Constructions

Q: How do these results hold up in other spaces? 2-D lattices.

Higher-D repeat patterns still have well-defined gradients in every direction:

Simply pick a basis set of directions, say $\vec{v}_1$ and $\vec{v}_2$ and define

$$F_\Theta(B) = \nabla^{\vec{v}_1}_\Theta (B) + \nabla^{\vec{v}_2}_\Theta (B)$$
Local Rule Constructions

Q: How do these results hold up in other spaces? 2-D lattices.

Higher-D repeat patterns still have well-defined gradients in every direction:

Simply pick a basis set of directions, say \( \vec{v}_1 \) and \( \vec{v}_2 \) and define

\[
F_\Theta(B) = \nabla_{\vec{v}_1}(B) + \nabla_{\vec{v}_2}(B)
\]
Q: How do these results hold up in other spaces?  2-D lattices.

Higher-D repeat patterns still have well-defined gradients in every direction:

Simply pick a basis set of directions, say \( \hat{v}_1 \) and \( \hat{v}_2 \) and define

\[
F_\Theta(B) = \nabla^{\hat{v}_1}_\Theta(B) + \nabla^{\hat{v}_2}_\Theta(B)
\]
**Definition.** A local check scheme is *single-choice* if the gradients along a vector basis are well-defined.
Local Rule Constructions

Q: How do these results hold up in other spaces? 2-D lattices.

**Definition.** A local check scheme is single-choice if the gradients along a vector basis are well-defined.
Local Rule Constructions

Q: How do these results hold up in other spaces? 2-D lattices.

**Definition.** A local check scheme is single-choice if the gradients along a vector basis are well-defined.

For the Sierpinski Gasket, the gradients are well-defined along the directions.
Local Rule Constructions

Q: How do these results hold up in other spaces? **2-D lattices.**

**Definition.** A *local check scheme* is **single-choice** if the gradients along a vector basis are well-defined.

For the Sierpinski Gasket, the gradients are well-defined along the directions so the gradient algorithm works.
Local Rule Constructions

Q: How do these results hold up in other spaces?  

**2-D lattices.**

**Definition.** A local check scheme is **single-choice** if the gradients along a vector basis are well-defined.

For the Sierpinski Gasket, the gradients are well-defined along the directions so the gradient algorithm works.
Local Rule Constructions

Q: How do these results hold up in other spaces? **2-D lattices.**

**Definition.** *A local check scheme is single-choice if the gradients along a vector basis are well-defined.*

For the Sierpinski Gasket, the gradients are well-defined along the directions so the gradient algorithm works.
Local Rule Constructions

Q: How do these results hold up in other spaces? 2-D lattices.

**Definition.** A local check scheme is single-choice if the gradients along a vector basis are well-defined.

For the Sierpinski Gasket, the gradients are well-defined along the directions so the gradient algorithm works.
Local Rule Constructions

Q: How do these results hold up in other spaces? 2-D lattices.

**Definition.** A local check scheme is single-choice if the gradients along a vector basis are well-defined.

For the Sierpinski Gasket, the gradients are well-defined along the directions so the gradient algorithm works.
Local Rule Constructions

Q: How do these results hold up in other spaces? 2-D lattices.

**Definition.** A local check scheme is **single-choice** if the gradients along a vector basis are well-defined.

For the Sierpinski Gasket, the gradients are well-defined along the directions so the gradient algorithm works.
Local Rule Constructions

Q: How do these results hold up in other spaces? **2-D lattices.**

**Definition.** A *local check scheme is single-choice* if the gradients along a vector basis are well-defined.

For the Sierpinski Gasket, the gradients are well-defined along the directions so the gradient algorithm works.

(Gradient of the check scheme is NOT well-defined along
Local Rule Constructions

Q: How do these results hold up in other spaces? 2-D lattices.

**Definition.** A local check scheme is single-choice if the gradients along a vector basis are well-defined.

For the Sierpinski Gasket, the gradients are well-defined along the directions

so the gradient algorithm works.

(Gradient of the check scheme is NOT well-defined along

so bases not always equivalent.)
Local Rule Constructions

Q: How do these results hold up in other spaces? 2-D lattices.

Just as certain more complicated check schemes in 1-D could be solved by combinations of gradients, so can be the 2-D “vector graphics” check schemes.
Local Rule Constructions

Q: How do these results hold up in other spaces? 2-D lattices.

Just as certain more complicated check schemes in 1-D could be solved by combinations of gradients, so can be the 2-D "vector graphics" check schemes.
Local Rule Constructions

Q: How do these results hold up in other spaces? 2-D lattices.

Just as certain more complicated check schemes in 1-D could be solved by combinations of gradients, so can be the 2-D ``vector graphics” check schemes.

\[ F = \nabla \vec{e}_1, -\vec{e}_2 \quad \text{Left-X} + \nabla -\vec{e}_1, +\vec{e}_2 \quad \text{Right-X} \]
Local Rule Constructions

Q: How do these results hold up in other spaces? 2-D lattices.

Just as certain more complicated check schemes in 1-D could be solved by combinations of gradients, so can be the 2-D "vector graphics" check schemes.

\[ F = \nabla \tilde{e}_1, -\tilde{e}_2 \text{ Left-}X + \nabla -\tilde{e}_1, +\tilde{e}_2 \text{ Right-}X \]
Local Rule Constructions

Q: How do these results hold up in other spaces? 2-D lattices.

Just as certain more complicated check schemes in 1-D could be solved by combinations of gradients, so can be the 2-D "vector graphics" check schemes.

\[ F = \nabla \xi_1, -\xi_2 \text{, Left-} X + \nabla -\xi_1, +\xi_2 \text{, Right-} X \]
Local Rule Constructions

Q: How do these results hold up in other spaces? **2-D lattices.**

Just as certain more complicated check schemes in 1-D could be solved by combinations of gradients, so can be the 2-D “vector graphics” check schemes.

\[ F = \nabla \xi_1, -\xi_2 \text{ Left-}X + \nabla -\xi_1, +\xi_2 \text{ Right-}X \]
Local Rule Constructions

Q: How do these results hold up in other spaces? 2-D lattices.

Just as certain more complicated check schemes in 1-D could be solved by combinations of gradients, so can be the 2-D “vector graphics” check schemes.

\[ F = \nabla \hat{\varepsilon}_1, -\hat{\varepsilon}_2 \]

\[ + \nabla -\hat{\varepsilon}_1, +\hat{\varepsilon}_2 \]

\[ + \sum_{i,j} \nabla \hat{\varepsilon}_i, \hat{\varepsilon}_j \]

\[ \text{Axis} \]
Local Rule Constructions

Q: How do these results hold up in other spaces? 2-D lattices.

Just as certain more complicated check schemes in 1-D could be solved by combinations of gradients, so can be the 2-D "vector graphics" check schemes.

\[ F = \nabla \vec{\xi}_1, -\vec{\xi}_2 \text{ Left-}X + \nabla -\vec{\xi}_1, +\vec{\xi}_2 \text{ Right-}X + \sum_{ij} \nabla \vec{\xi}_i, \vec{\xi}_j \text{ Axis} \]
Local Rule Constructions

Q: How do these results hold up in other spaces? 2-D lattices.

Just as certain more complicated check schemes in 1-D could be solved by combinations of gradients, so can be the 2-D “vector graphics” check schemes.

\[
F = \nabla \tilde{e}_1, -\tilde{e}_2 \quad \text{Left-X} + \nabla -\tilde{e}_1, +\tilde{e}_2 \quad \text{Right-X} + \sum_{ij} \nabla \tilde{e}_i, \tilde{e}_j \quad \text{Axis} + \nabla -\tilde{e}_1, -\tilde{e}_2 \quad \text{Left-Green} + \nabla +\tilde{e}_1, -\tilde{e}_2 \quad \text{Left-Green} + \nabla +\tilde{e}_1, -\tilde{e}_2 \quad \text{Right-Red} + \nabla +\tilde{e}_1, +\tilde{e}_2 \quad \text{Right-Red}
\]
Local Rule Constructions

Q: How do these results hold up in other spaces? 2-D lattices.

Just as certain more complicated check schemes in 1-D could be solved by combinations of gradients, so can be the 2-D “vector graphics” check schemes.

\[ F = \nabla \tilde{e}_1, -\tilde{e}_2 \quad \text{Left-X} + \nabla -\tilde{e}_1, +\tilde{e}_2 \quad \text{Right-X} \]

\[ + \sum_{ij} \nabla \tilde{e}_i, \tilde{e}_j \quad \text{Axis} \]

\[ + \nabla -\tilde{e}_1, -\tilde{e}_2 \quad \text{Left-Green} + \nabla +\tilde{e}_1, -\tilde{e}_2 \quad \text{Left-Green} \]

\[ + \nabla +\tilde{e}_1, -\tilde{e}_2 \quad \text{Right-Red} + \nabla +\tilde{e}_1, +\tilde{e}_2 \quad \text{Right-Red} \]

(+ b’dry reflection rules)
Local Rule Constructions

Q: How do these results hold up in other spaces? 2-D lattices.

Just as certain more complicated check schemes in 1-D could be solved by combinations of gradients, so can be the 2-D "vector graphics" check schemes.

\[ F = \nabla \varepsilon_1, -\varepsilon_2_{\text{Left--X}} + \nabla -\varepsilon_1, +\varepsilon_2_{\text{Right--X}} + \sum_{ij} \nabla \varepsilon_i, \varepsilon_j_{\text{Axis}} + \nabla -\varepsilon_1, -\varepsilon_2_{\text{Left--Green}} + \nabla +\varepsilon_1, -\varepsilon_2_{\text{Left--Green}} + \nabla +\varepsilon_1, -\varepsilon_2_{\text{Right--Red}} + \nabla +\varepsilon_1, +\varepsilon_2_{\text{Right--Red}} (+ \text{ b’dry reflection rules}) \]

Required state and radius is bounded by sum over components.
Local Rule Constructions

Q: How do these results hold up in other spaces? 2-D lattices.
Local Rule Constructions

Q: How do these results hold up in other spaces? 2-D lattices.

The “Naive Backtracking” algorithm also generalizes.
Local Rule Constructions

Q: How do these results hold up in other spaces? 2-D lattices.

The “Naive Backtracking” algorithm also generalizes.

Given a basepoint,
Local Rule Constructions

Q: How do these results hold up in other spaces?  2-D lattices.

The “Naive Backtracking” algorithm also generalizes.

Given a basepoint, consider the “standard spiral path.”
Local Rule Constructions

Q: How do these results hold up in other spaces? 2-D lattices.

The “Naive Backtracking” algorithm also generalizes.

Given a basepoint, consider the “standard spiral path.”
Local Rule Constructions

The technique is to implement Naive Backtracking along the standard path, which is one-dimensional.
Local Rule Constructions

The technique is to implement Naive Backtracking along the standard path, which is one-dimensional.
Local Rule Constructions

The technique is to implement Naive Backtracking *along* the standard path, which *is* one-dimensional.
Local Rule Constructions

The technique is to implement Naive Backtracking *along* the standard path, which *is* one-dimensional.
Local Rule Constructions

The technique is to implement Naive Backtracking along the standard path, which is one-dimensional.
Local Rule Constructions

The technique is to implement Naive Backtracking *along* the standard path, which *is* one-dimensional.
Local Rule Constructions

The technique is to implement Naive Backtracking *along* the standard path, which *is* one-dimensional.
Local Rule Constructions

The technique is to implement Naive Backtracking *along* the standard path, which *is* one-dimensional.
Local Rule Constructions

The technique is to implement Naive Backtracking along the standard path, which is one-dimensional.
Local Rule Constructions

The technique is to implement Naive Backtracking *along* the standard path, which *is* one-dimensional.

The virtual Turing heads -- which are 0-dimensional points -- move along a 1-dimensional path.
Local Rule Constructions

The technique is to implement Naive Backtracking \textit{along} the standard path, which \textit{is} one-dimensional.

The virtual Turing heads -- which are 0-dimensional points -- move along a 1-dimensional path.

They create an expanding and contracting spiral.
Local Rule Constructions

The technique is to implement Naive Backtracking along the standard path, which is one-dimensional.

The virtual Turing heads -- which are 0-dimensional points -- move along a 1-dimensional path.

They create an expanding and contracting spiral.
Local Rule Constructions

- "Border of correctness" relative to the local check scheme expands *outwards* and contracts *inwards*

The virtual Turing heads -- which are 0-dimensional points -- move along a 1-dimensional path.

They create an expanding and contracting spiral.
Local Rule Constructions

- “Border of correctness” relative to the local check scheme expands *outwards* and contracts *inwards*

The virtual Turing heads -- which are 0-dimensional points -- move along a 1-dimensional path.

They create an expanding and contracting spiral.

There are some technicalities of setting local rules to “protect” the border of the spiral.
Local Rule Constructions
Local Rule Constructions

The construction of a spiral “naive backtracking rule” $F_{\Theta}$ shows:

**Theorem.** Local checkability is a necessary and sufficient condition for robust solvability in euclidean lattices; and any solvable pattern $T$ has a solution $F$ with

$$r(F) \leq 4LCR(T) + 8.$$
Local Rule Constructions

The construction of a spiral “naive backtracking rule” $F_\Theta$ shows:

**Theorem.** Local checkability is a necessary and sufficient condition for robust solvability in euclidean lattices; and any solvable pattern $T$ has a solution $F$ with

$$r(F) \leq 4LCR(T) + 8.$$ 

Architecturally:
Local Rule Constructions

The construction of a spiral “naive backtracking rule” $F_{\Theta}$ shows:

**Theorem.** Local checkability is a necessary and sufficient condition for robust solvability in euclidean lattices; and any solvable pattern $T$ has a solution $F$ with

$$r(F) \leq 4LCR(T) + 8.$$ 

Architecturally:

- Single-choice check schemes have coherent $(n-1)$-dimensional gradient solutions in an $n$-dimensional space (fast).
Local Rule Constructions

The construction of a spiral “naive backtracking rule” \( F_\Theta \) shows:

**Theorem.** *Local checkability is a necessary and sufficient condition for robust solvability in euclidean lattices; and any solvable pattern \( T \) has a solution \( F \) with*

\[
r(F) \leq 4LCR(T) + 8.
\]

**Architecturally:**

- Single-choice check schemes have coherent \((n-1)\)-dimensional gradient solutions in an \( n \)-dimensional space (fast).
- Special forms of composite check schemes have multi-gradient solutions (fast).
Local Rule Constructions

The construction of a spiral "naive backtracking rule" $F_\Theta$ shows:

**Theorem.** Local checkability is a necessary and sufficient condition for robust solvability in euclidean lattices; and any solvable pattern $T$ has a solution $F$ with

$$r(F) \leq 4LCR(T) + 8.$$

Architecturally:

- Single-choice check schemes have coherent $(n-1)$-dimensional gradient solutions in an $n$-dimensional space (fast).
- Special forms of composite check schemes have multi-gradient solutions (fast).
- Arbitrary check schemes can be solved by a 0-dimensional Turing-head "points" moving in $n$-dimensional space (may be very slow).
Local Rule Constructions

The construction of a spiral “naive backtracking rule” $F_{\Theta}$ shows:

**Theorem.** Local checkability is a necessary and sufficient condition for robust solvability in euclidean lattices; and any solvable pattern $T$ has a solution $F$ with

$$r(F) \leq 4LCR(T) + 8.$$ 

Architecturally:

- Single-choice check schemes have coherent (n-1)-dimensional gradient solutions in an n-dimensional space (fast).
- Special forms of composite check schemes have multi-gradient solutions (fast).
- Arbitrary check schemes can be solved by a 0-dimensional Turing-head “points” moving in n-dimensional space (may be very slow).

**Proposition.** *Every euclidean check scheme $\Theta$ has a “smart” solution $\widehat{F}_{\Theta}$ solving $\Theta$ with worst-case runtime $O(n)$.*
Local Rule Constructions

The construction of a spiral “naive backtracking rule” $F_{\Theta}$ shows:

**Theorem.** *Local checkability is a necessary and sufficient condition for robust solvability in euclidean lattices; and any solvable pattern $T$ has a solution $F$ with*

$$r(F) \leq 4LCR(T) + 8.$$  

**Architecturally:**

- Single-choice check schemes have coherent (n-1)-dimensional gradient solutions in an n-dimensional space (fast).
- Special forms of composite check schemes have multi-gradient solutions (fast).
- Arbitrary check schemes can be solved by a 0-dimensional Turing-head “points” moving in n-dimensional space (may be very slow).

**Proposition.** *Every euclidean check scheme $\Theta$ has a “smart” solution $\bar{F}_{\Theta}$ solving $\Theta$ with worst-case runtime $O(n)$.*
Local Rule Constructions

These constructions are the next step of a “global-to-local” compiler:
Local Rule Constructions

These constructions are the next step of a “global-to-local” compiler:

Input: Pattern
Local Rule Constructions

These constructions are the next step of a “global-to-local” compiler:

Input: Pattern

Step 1: Local checkability

- min radius = 2 (3 states) \( \Theta_{Stripe} \)
- min radius = 1 (2 states) \( \Theta_{Watermelon} \)
- min radius = 2 (2 states) \( \Theta_{Cross} \)
- min radius = 4 (3 states) \( \Theta_{DV-Split} \)
Local Rule Constructions

These constructions are the next step of a “global-to-local” compiler:

Input: Pattern

Step 1: Local checkability

- min radius = 2 (3 states) \( \Theta_{Stripe} \)
- min radius = 1 (2 states) \( \Theta_{Watermelon} \)
- min radius = 2 (2 states) \( \Theta_{Cross} \)
- min radius = 4 (3 states) \( \Theta_{DV-Split} \)

Step 2: Multi-gradient construction
Local Rule Constructions

These constructions are the next step of a “global-to-local” compiler:

**Input: Pattern**

![Pattern Diagrams]

**Step 1: Local checkability**

- min radius = 2
  - (3 states)
  - Θ_{Stripe}
- min radius = 1
  - (2 states)
  - Θ_{Watermelon}
- min radius = 2
  - (2 states)
  - Θ_{Cross}
- min radius = 4
  - (3 states)
  - Θ_{DV-Split}

**Step 2: Multi-gradient construction**

- stripe_rule
- watermelon_rule
- cross_rule
- DV-split_rule