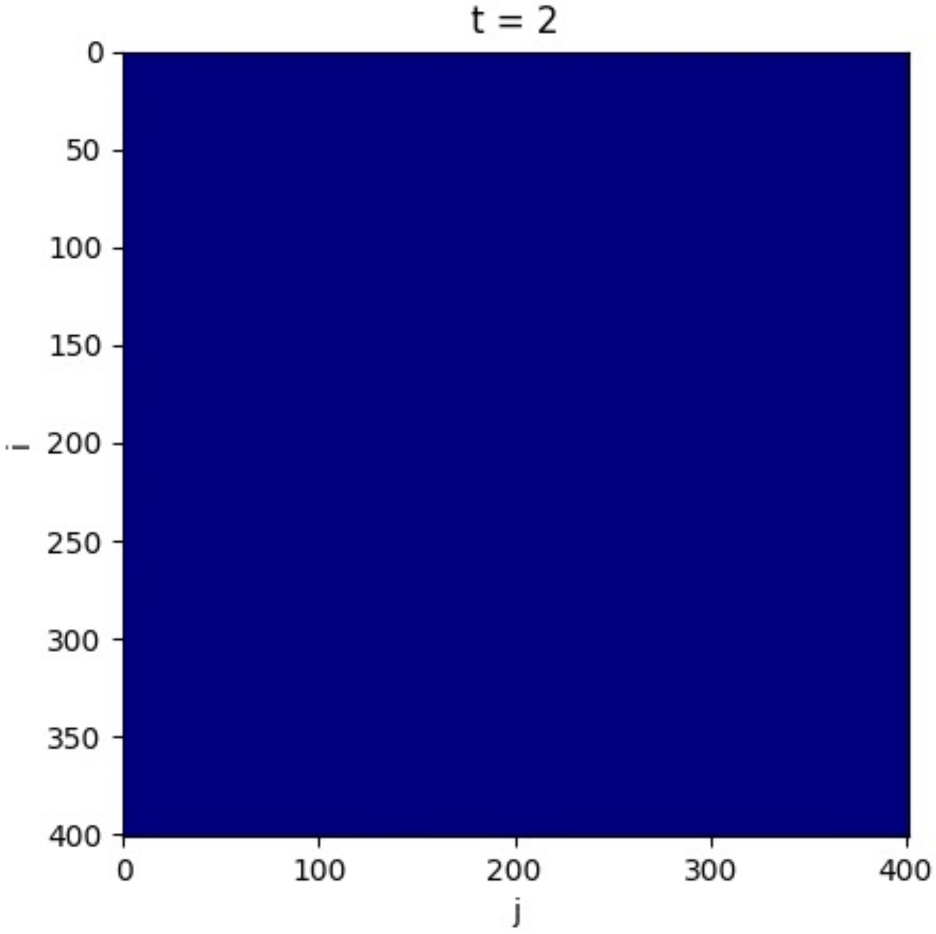
IMPACT 2023

Automatic Algorithm-Based Fault Detection (AABFD) of Stencil Computations

Louis Narmour (UR1+CSU), Steven Derrien (UR1), Sanjay Rajopadhye (CSU)

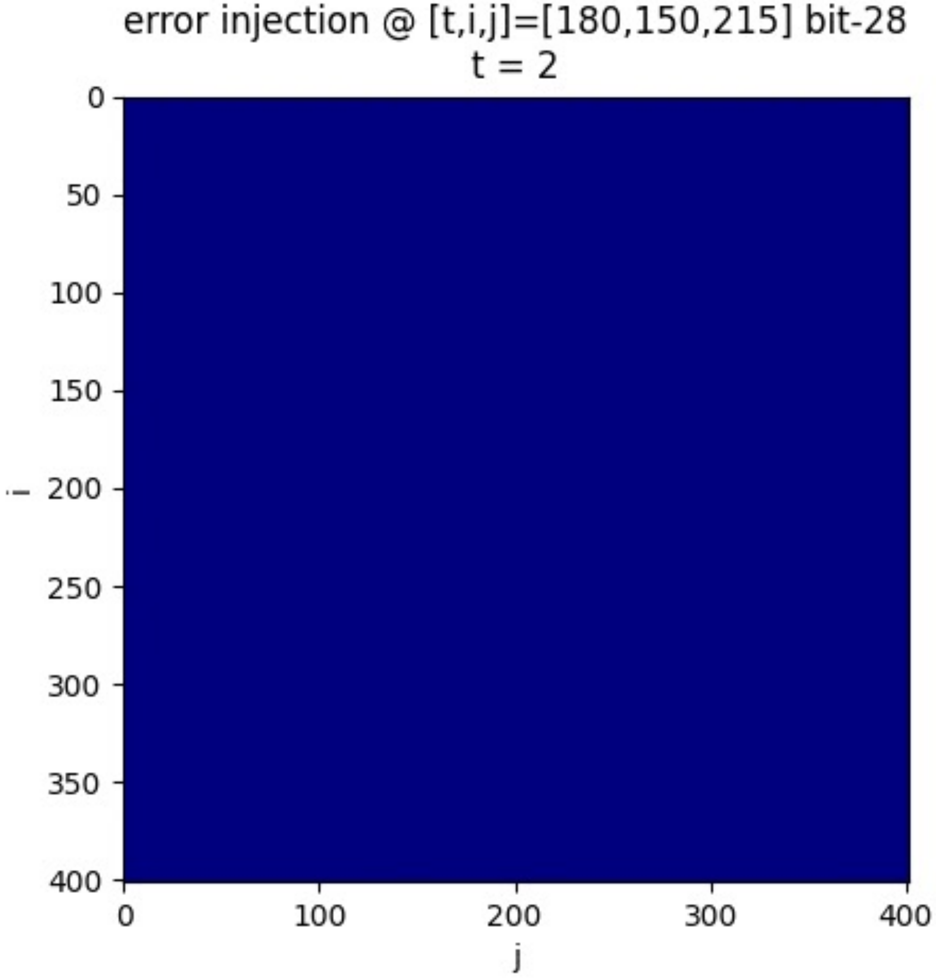
Stencils are commonly used to compute approximate solutions to partial differential equations modeling physical phenomena like wave propagation

$$X_{i,j}^{t} = \begin{cases} I_{i,j} & t = 0 \\ w_{i-1,j}X_{i-1,j}^{t-1} + w_{i,j-1}X_{i,j-1}^{t-1} + w_{i,j}X_{i,j}^{t-1} + w_{i,j+1}X_{i,j+1}^{t-1} + w_{i+1,j}X_{i+1,j}^{t-1} & t > 0 \end{cases}$$



Transient silent errors are difficult to handle because they manifest as data corruption

$$X_{i,j}^{t} = \begin{cases} I_{i,j} & t = 0 \\ \begin{matrix} w_{i-1,j}X_{i-1,j}^{t-1} \\ w_{i,j-1}X_{i,j-1}^{t-1} + w_{i,j}X_{i,j}^{t-1} + w_{i,j+1}X_{i,j+1}^{t-1} + \\ w_{i+1,j}X_{i+1,j}^{t-1} \end{matrix} & t > 0 \end{cases}$$



What is this about?

What: **Hardware error detection technique** at the software level

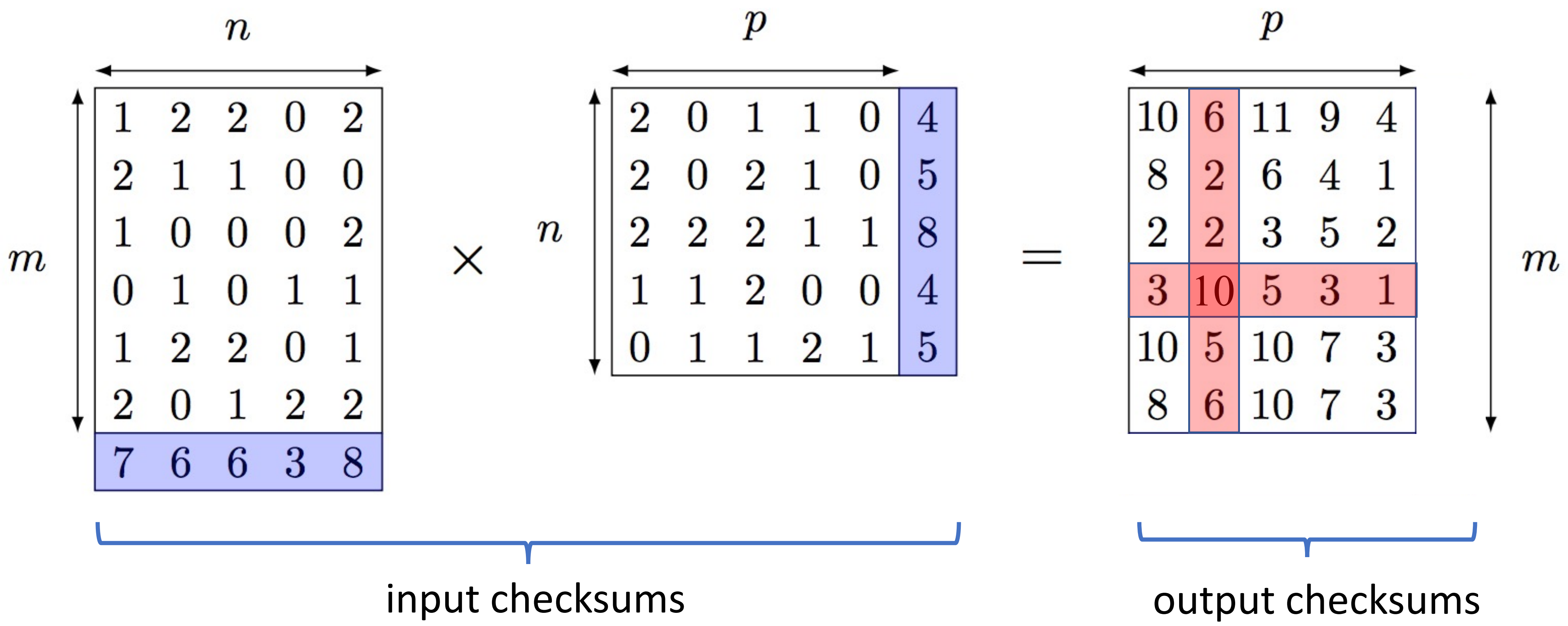
How: Based on **algorithmic** and **algebraic** level properties

Where: Application-specific (for stencils)

Key insight: **Can be automated** thanks to polyhedral compilation

Stencil program ***susceptible*** to transient silent errors → Polyhedral compilation & analysis → Stencil program ***resiliant*** to transient silent errors

Algorithm-based fault tolerance (ABFT) for matrix product

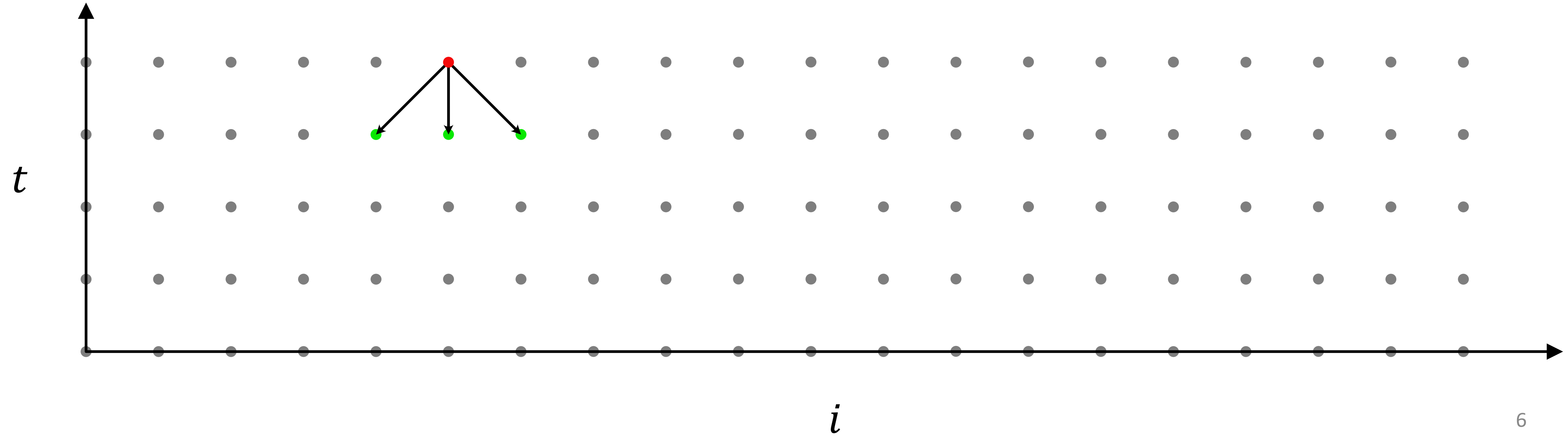


Kuang-Hua Huang and Jacob A. Abraham. *Algorithm-based fault tolerance for matrix operations*. 1984, IEEE transactions on computers.

How does one *do ABFT* on stencils?

Consider the simplest 1D Jacobi example:

$$X_{t,i} = \begin{cases} I_i & t = 0 \\ X_{t-1,i} & 0 < t \leq T \text{ and } (i = 0 \text{ or } i = N) \\ w_0 X_{t-1,i-1} + w_1 X_{t-1,i} + w_2 X_{t-1,i+1} & \text{otherwise} \end{cases}$$

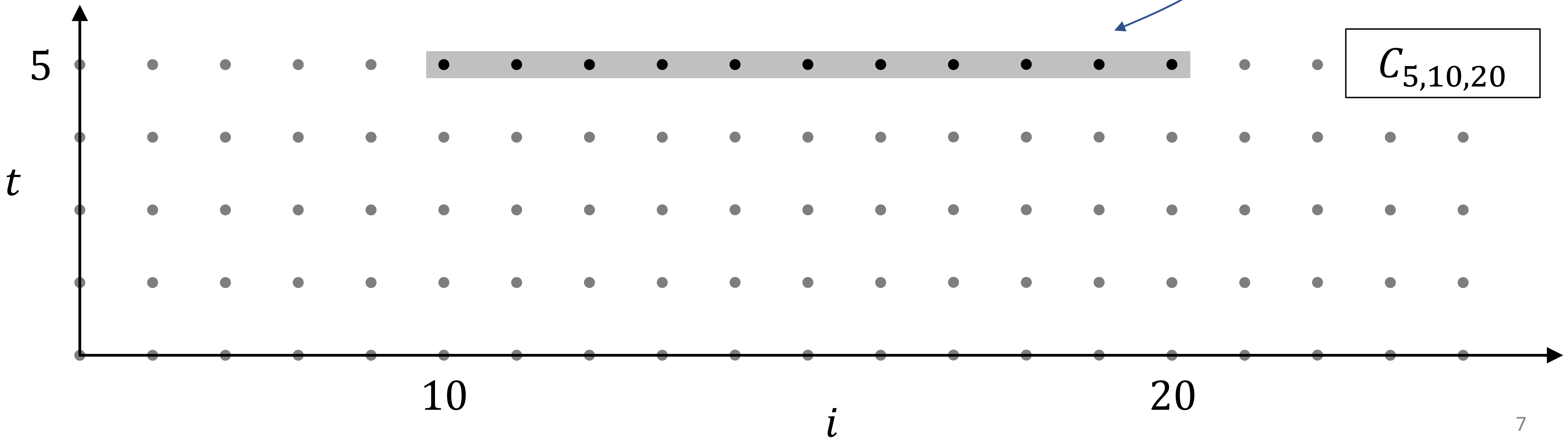


Let C be the checksum over some window of values

$$C_{t,l,m} = \sum_{i=l}^{m} X_{t,i}$$

$$C_{5,10,20} = \sum_{i=10}^{20} X_{5,i}$$

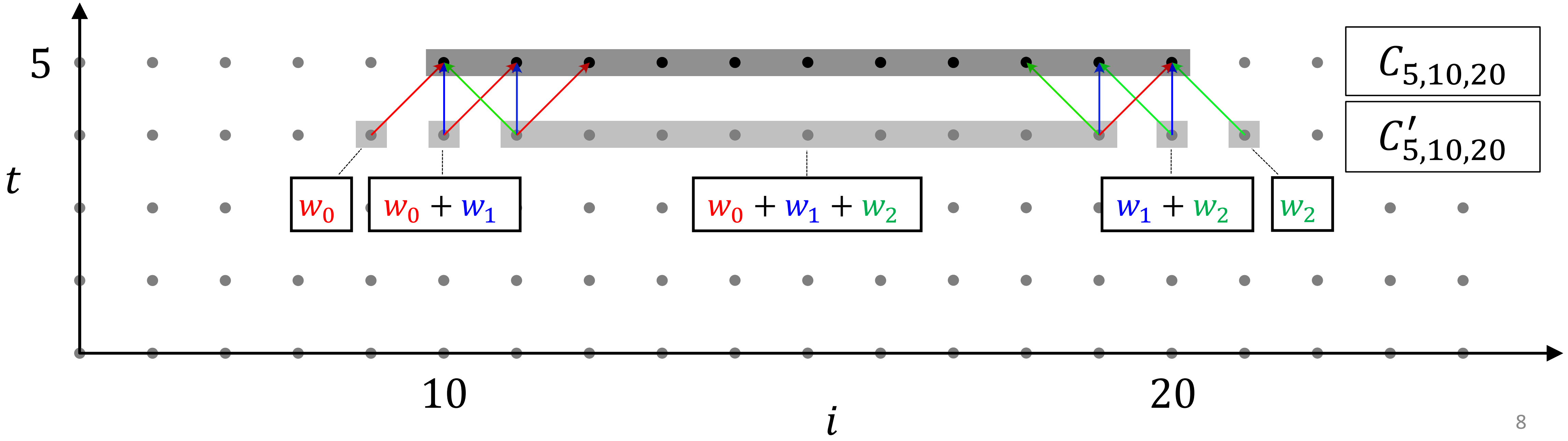
i.e., the sum of these elements



Let C' be an ***algebraically equivalent*** expression

$$C'_{t,l,m} = \sum_{i=l}^{m} \left(w_0 X_{t-1,i-1} + w_1 X_{t-1,i} + w_2 X_{t-1,i+1}\right)$$

$$C'_{5,10,20} = (w_0)X_{4,9} + (w_0 + w_1)X_{4,10} + (w_0 + w_1 + w_2)\sum_{i=11}^{19} X_{4,i} + (w_1 + w_2)X_{4,20} + (w_2)X_{4,21}$$

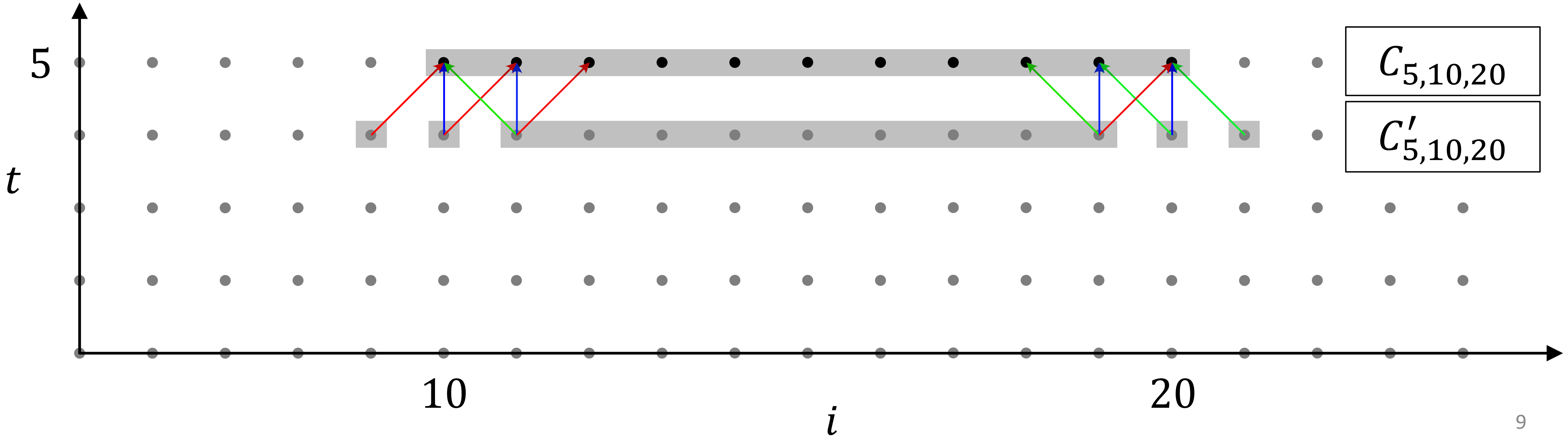


Errors manifest as a large difference between C and C'

$$\Delta C_{t,l,m} \equiv \left| C_{t,l,m} - C'_{t,l,m} \right|$$

$$\Delta C_{t,l,m} > \text{threshold}$$

if observed then we have detected an error

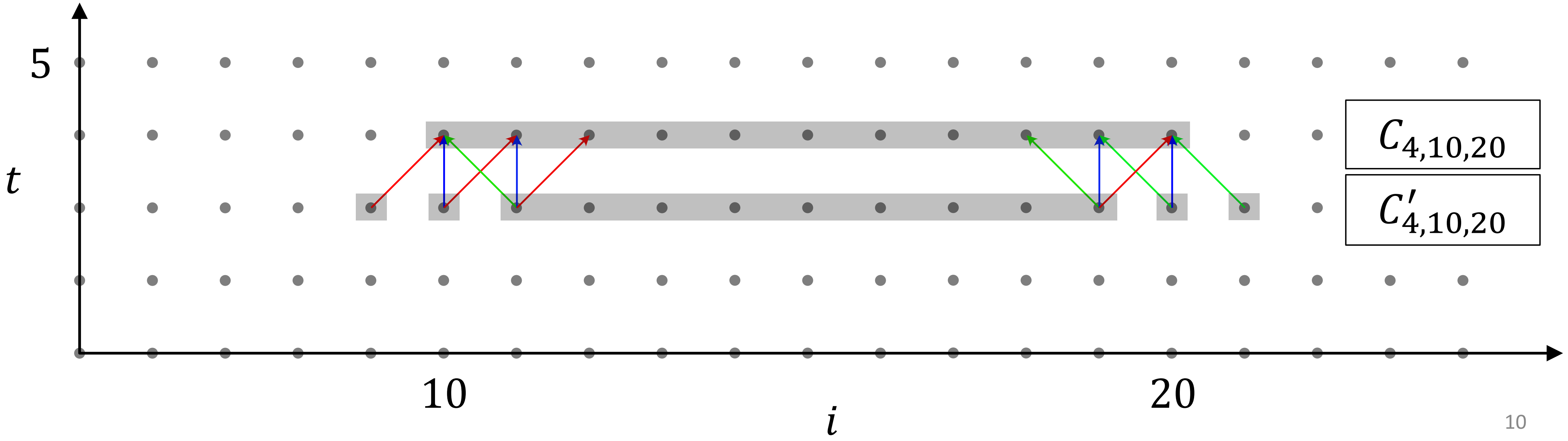


Errors manifest as a large difference between C and C'

$$\Delta C_{t,l,m} \equiv \left| C_{t,l,m} - C'_{t,l,m} \right|$$

$$\Delta C_{t,l,m} > \text{threshold}$$

if observed then we have detected an error

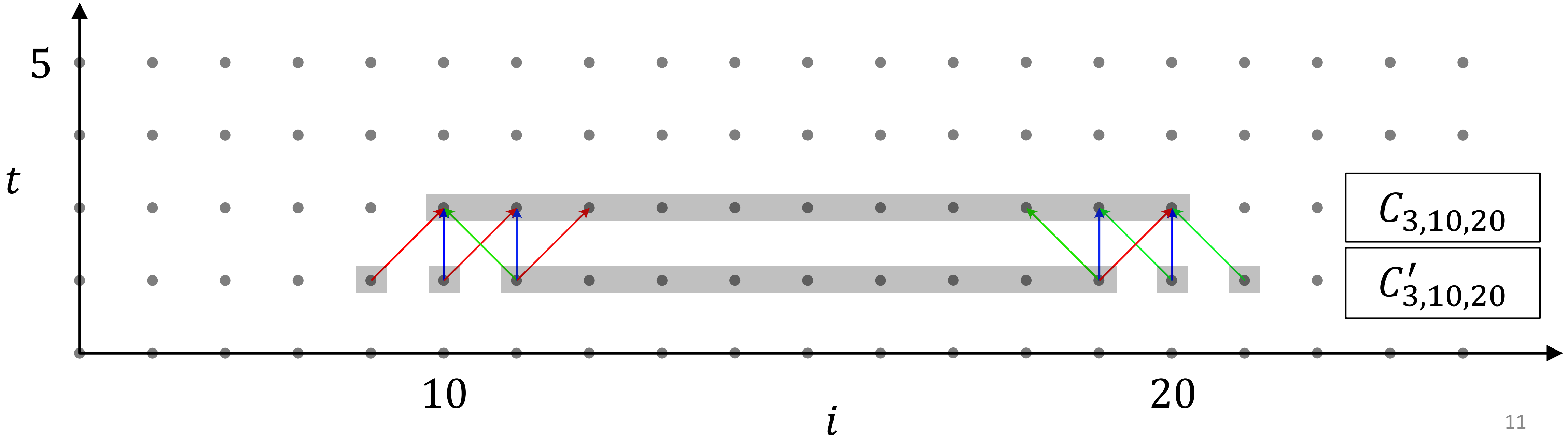


Errors manifest as a large difference between C and C'

$$\Delta C_{t,l,m} \equiv |C_{t,l,m} - C'_{t,l,m}|$$

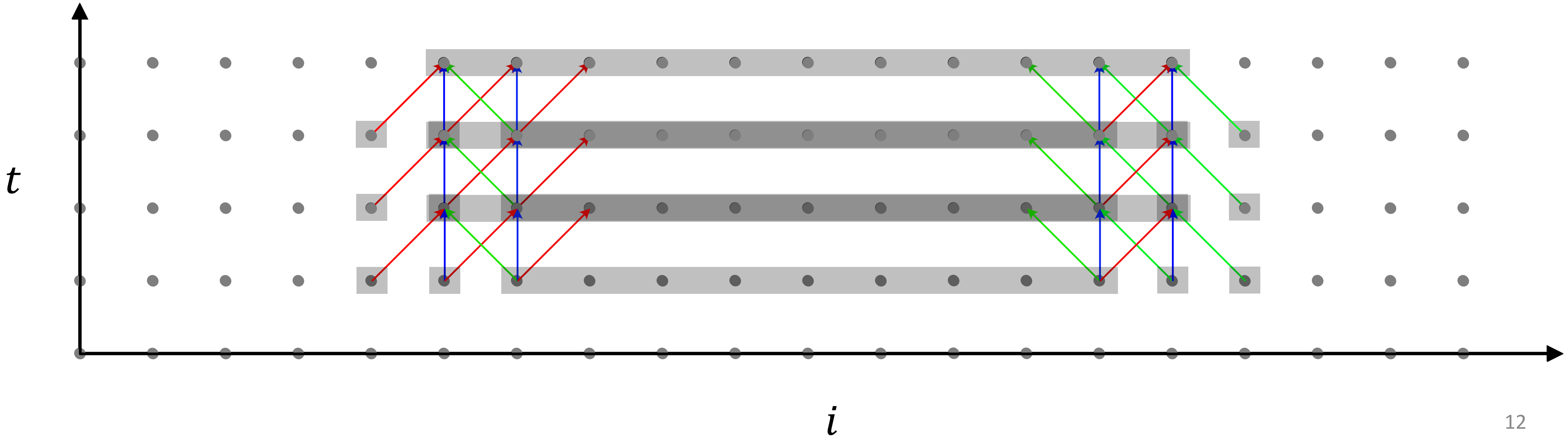
$$\Delta C_{t,l,m} > \text{threshold}$$

if observed then we have detected an error



Checking ΔC across **every time step** works, but is **inefficient**

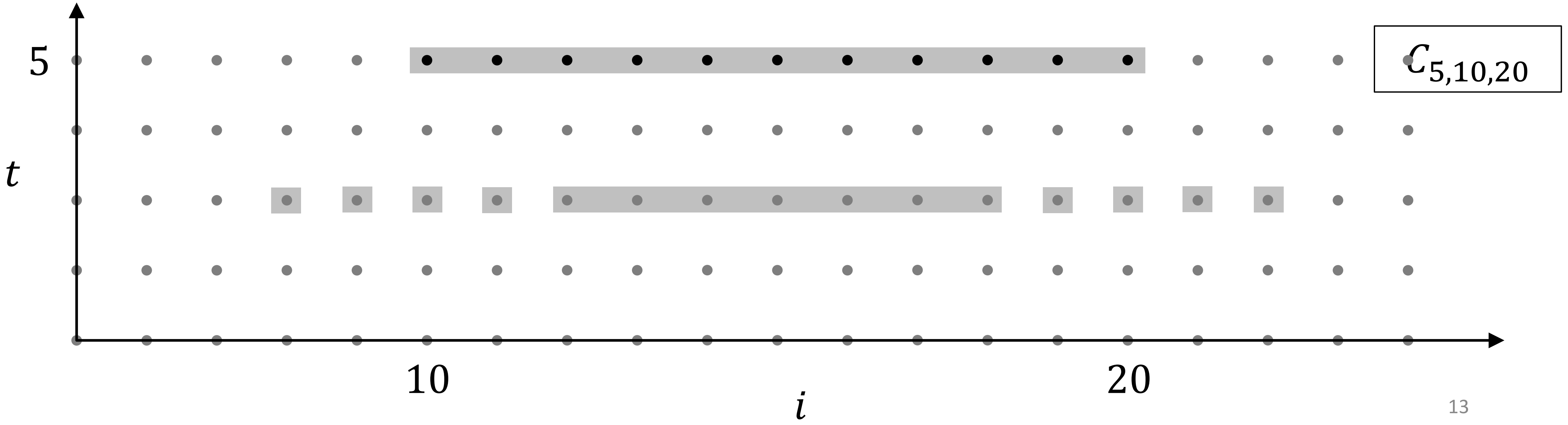
- Across successive timesteps, work is asymptotically same as the stencil
- At each time step, two "sweeps" across i are needed (for overlapping C and C')
- Can we be more efficient?



Construct C' using two substitutions

$$C'_{t,l,m} = \sum_{i=l}^{m} \left(w_0 X_{t-1,i-1} + w_1 X_{t-1,i} + w_2 X_{t-1,i+1} \right)$$

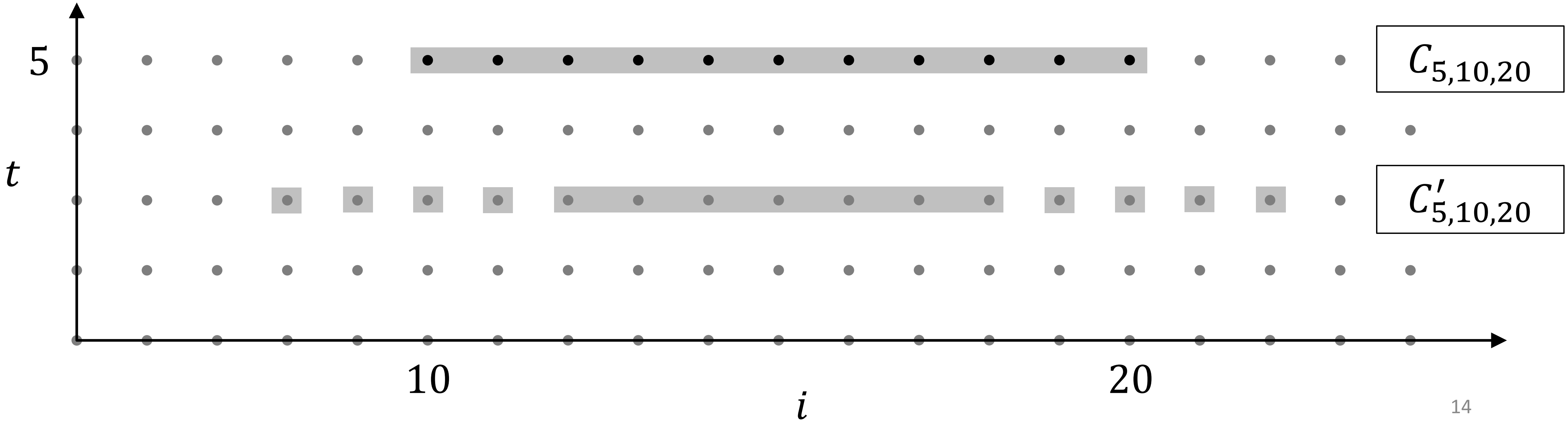
$$C'_{t,l,m} = \sum_{i=l}^{m} \left(\begin{array}{r} w_0 \left(w_0 X_{t-2,i-2} + w_1 X_{t-2,i-1} + w_2 X_{t-2,i} \right) + \\ w_1 \left(w_0 X_{t-2,i-1} + w_1 X_{t-2,i} + w_2 X_{t-2,i+1} \right) \\ w_2 \left(w_0 X_{t-2,i} + w_1 X_{t-2,i+1} + w_2 X_{t-2,i+2} \right) \end{array} \right)$$



Construct C' using two substitutions

$$C'_{t,l,m} = \sum_{i=l}^{m} \begin{pmatrix} w_0 (w_0 X_{t-2,i-2} + w_1 X_{t-2,i-1} + w_2 X_{t-2,i}) + \\ w_1 (w_0 X_{t-2,i-1} + w_1 X_{t-2,i} + w_2 X_{t-2,i+1}) \\ w_2 (w_0 X_{t-2,i} + w_1 X_{t-2,i+1} + w_2 X_{t-2,i+2}) \end{pmatrix}$$

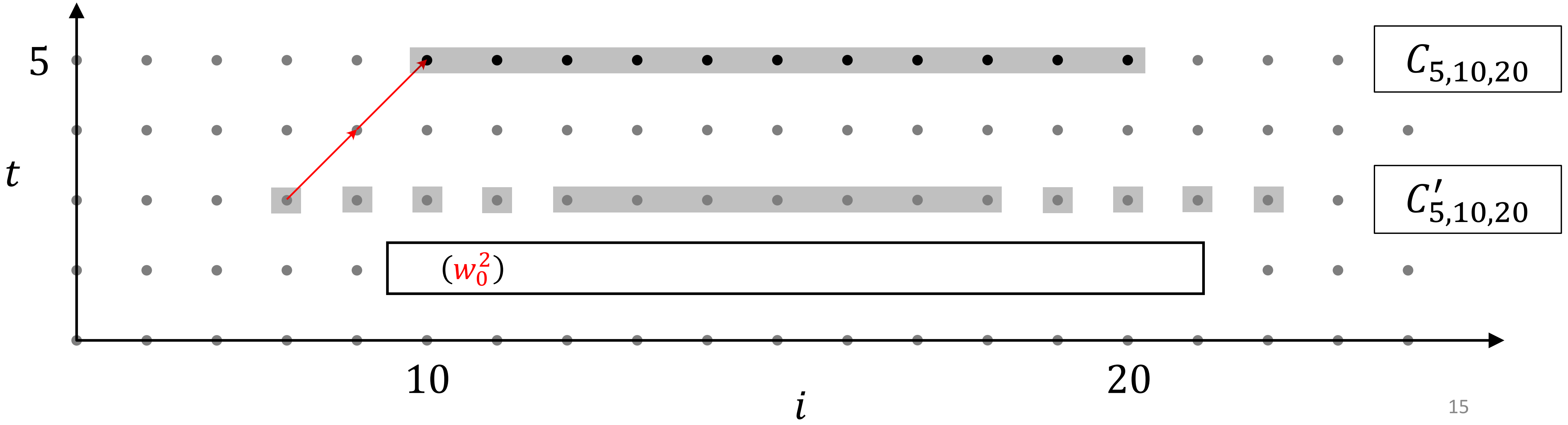
$$C'_{5,10,20} = (\dots)X_{3,8} + (\dots)X_{3,9} + (\dots)X_{3,10} + (\dots)X_{3,11} + (\dots)\sum_{i=12}^{18} X_{3,i} + (\dots)X_{3,19} + (\dots)X_{3,20} + (\dots)X_{3,21} + (\dots)X_{3,22}$$



Construct C' using two substitutions

$$C'_{t,l,m} = \sum_{i=l}^{m} \left(\begin{array}{r} w_0\left(w_0 X_{t-2,i-2} + w_1 X_{t-2,i-1} + w_2 X_{t-2,i}\right) + \\ w_1\left(w_0 X_{t-2,i-1} + w_1 X_{t-2,i} + w_2 X_{t-2,i+1}\right) \\ w_2\left(w_0 X_{t-2,i} + w_1 X_{t-2,i+1} + w_2 X_{t-2,i+2}\right) \end{array} \right)$$

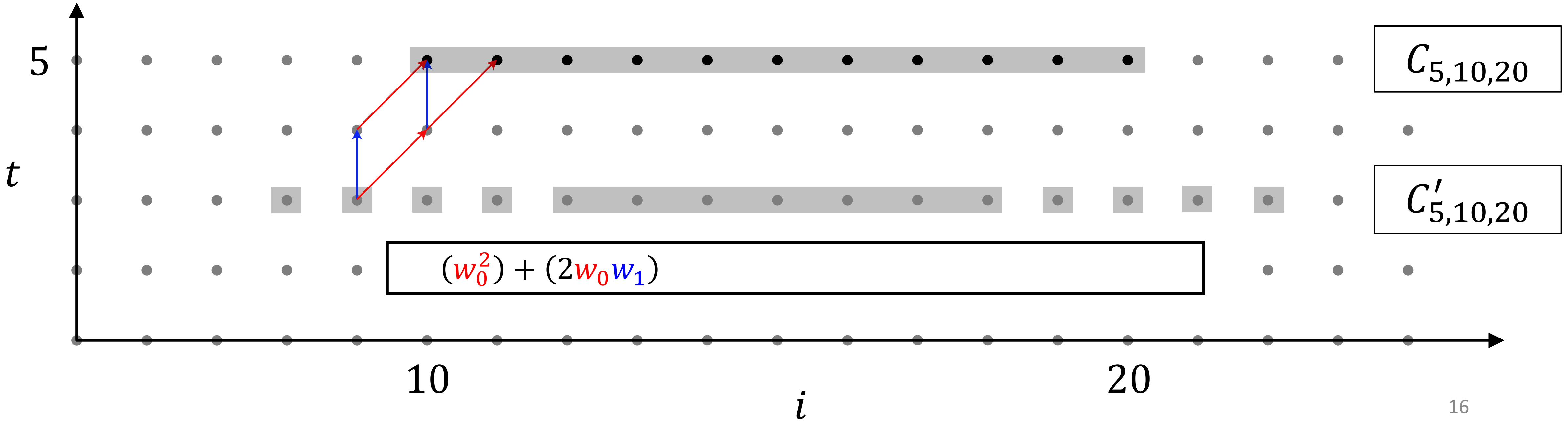
$$C'_{5,10,20} = (\dots)X_{3,8} + (\dots)X_{3,9} + (\dots)X_{3,10} + (\dots)X_{3,11} + (\dots)\sum_{i=12}^{18} X_{3,i} + (\dots)X_{3,19} + (\dots)X_{3,20} + (\dots)X_{3,21} + (\dots)X_{3,22}$$



Construct C' using two substitutions

$$C'_{t,l,m} = \sum_{i=l}^{m} \begin{pmatrix} w_0\left(w_0 X_{t-2,i-2} + w_1 X_{t-2,i-1} + w_2 X_{t-2,i}\right) + \\ w_1\left(w_0 X_{t-2,i-1} + w_1 X_{t-2,i} + w_2 X_{t-2,i+1}\right) \\ w_2\left(w_0 X_{t-2,i} + w_1 X_{t-2,i+1} + w_2 X_{t-2,i+2}\right) \end{pmatrix}$$

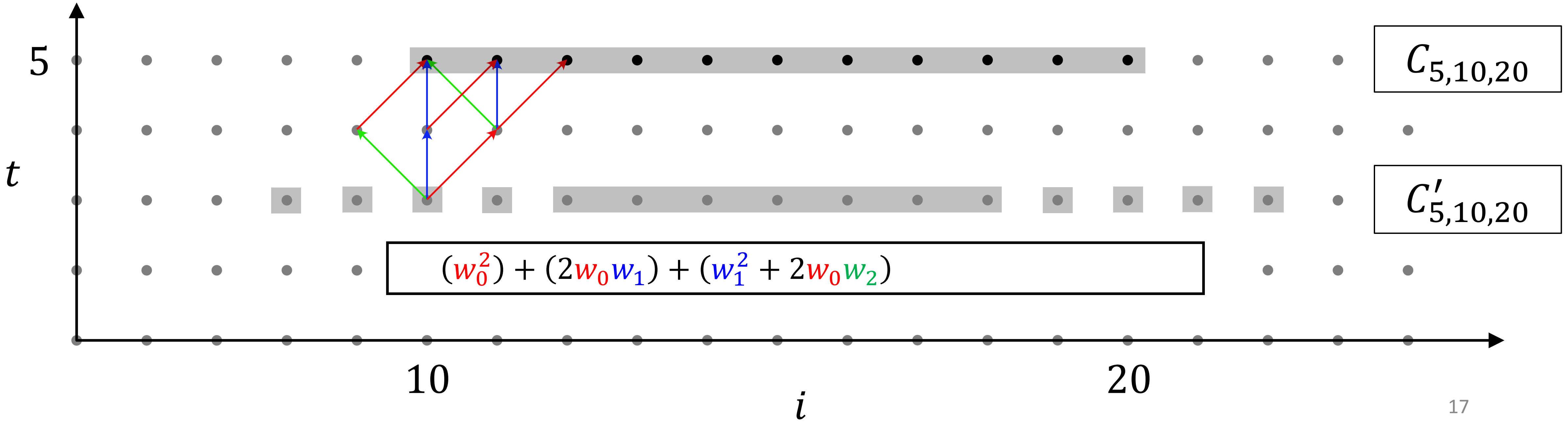
$$C'_{5,10,20} = (\dots)X_{3,8} + (\dots)X_{3,9} + (\dots)X_{3,10} + (\dots)X_{3,11} + (\dots)\sum_{i=12}^{18} X_{3,i} + (\dots)X_{3,19} + (\dots)X_{3,20} + (\dots)X_{3,21} + (\dots)X_{3,22}$$



Construct C' using two substitutions

$$C'_{t,l,m} = \sum_{i=l}^{m} \left(\begin{array}{r} w_0\left(w_0 X_{t-2,i-2} + w_1 X_{t-2,i-1} + w_2 X_{t-2,i}\right) + \\ w_1\left(w_0 X_{t-2,i-1} + w_1 X_{t-2,i} + w_2 X_{t-2,i+1}\right) \\ w_2\left(w_0 X_{t-2,i} + w_1 X_{t-2,i+1} + w_2 X_{t-2,i+2}\right) \end{array} \right)$$

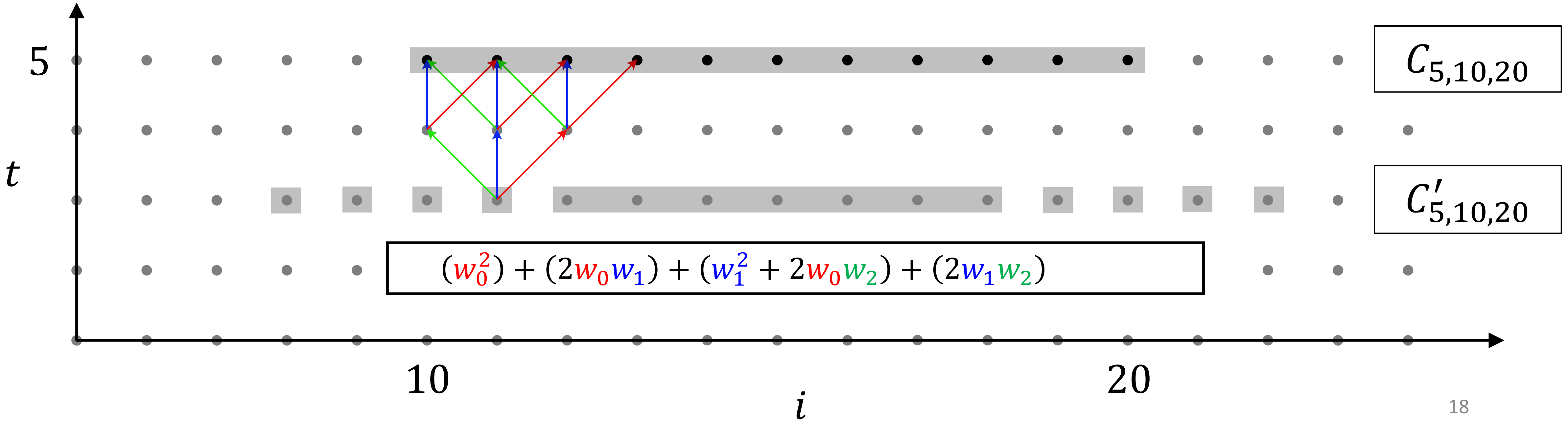
$$C'_{5,10,20} = (\dots)X_{3,8} + (\dots)X_{3,9} + (\dots)X_{3,10} + (\dots)X_{3,11} + (\dots)\sum_{i=12}^{18} X_{3,i} + (\dots)X_{3,19} + (\dots)X_{3,20} + (\dots)X_{3,21} + (\dots)X_{3,22}$$



Construct C' using two substitutions

$$C'_{t,l,m} = \sum_{i=l}^{m} \begin{pmatrix} w_0\left(w_0 X_{t-2,i-2} + w_1 X_{t-2,i-1} + w_2 X_{t-2,i}\right) + \\ w_1\left(w_0 X_{t-2,i-1} + w_1 X_{t-2,i} + w_2 X_{t-2,i+1}\right) \\ w_2\left(w_0 X_{t-2,i} + w_1 X_{t-2,i+1} + w_2 X_{t-2,i+2}\right) \end{pmatrix}$$

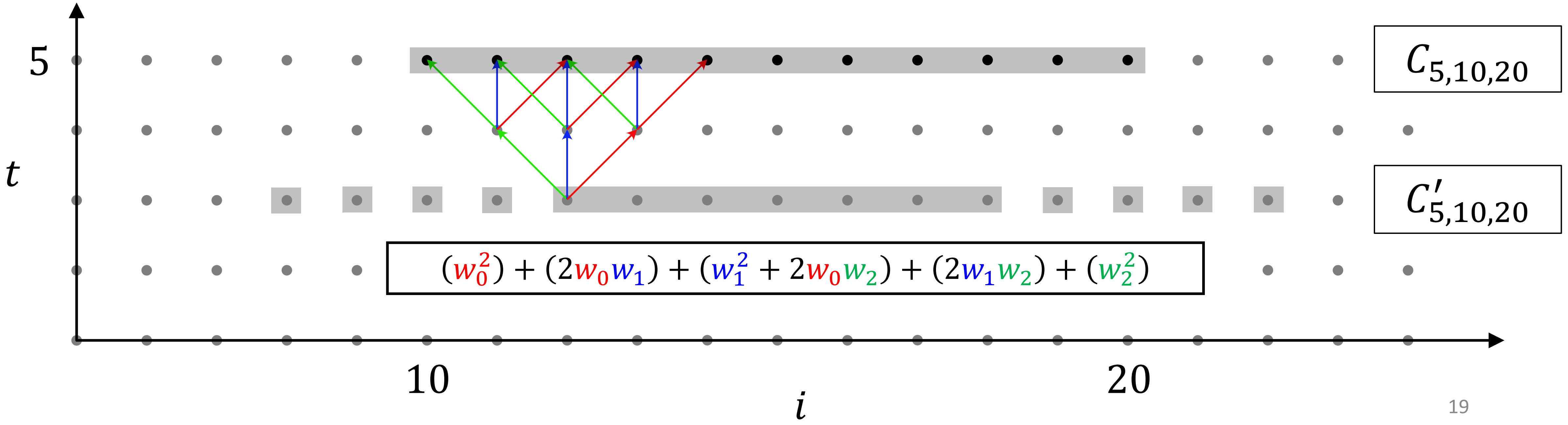
$$C'_{5,10,20} = (\ldots)X_{3,8} + (\ldots)X_{3,9} + (\ldots)X_{3,10} + (\ldots)X_{3,11} + (\ldots)\sum_{i=12}^{18} X_{3,i} + (\ldots)X_{3,19} + (\ldots)X_{3,20} + (\ldots)X_{3,21} + (\ldots)X_{3,22}$$



Construct C' using two substitutions

$$C'_{t,l,m} = \sum_{i=l}^{m} \left(\begin{array}{r} w_0\left(w_0 X_{t-2,i-2} + w_1 X_{t-2,i-1} + w_2 X_{t-2,i}\right) + \\ w_1\left(w_0 X_{t-2,i-1} + w_1 X_{t-2,i} + w_2 X_{t-2,i+1}\right) \\ w_2\left(w_0 X_{t-2,i} + w_1 X_{t-2,i+1} + w_2 X_{t-2,i+2}\right) \end{array} \right)$$

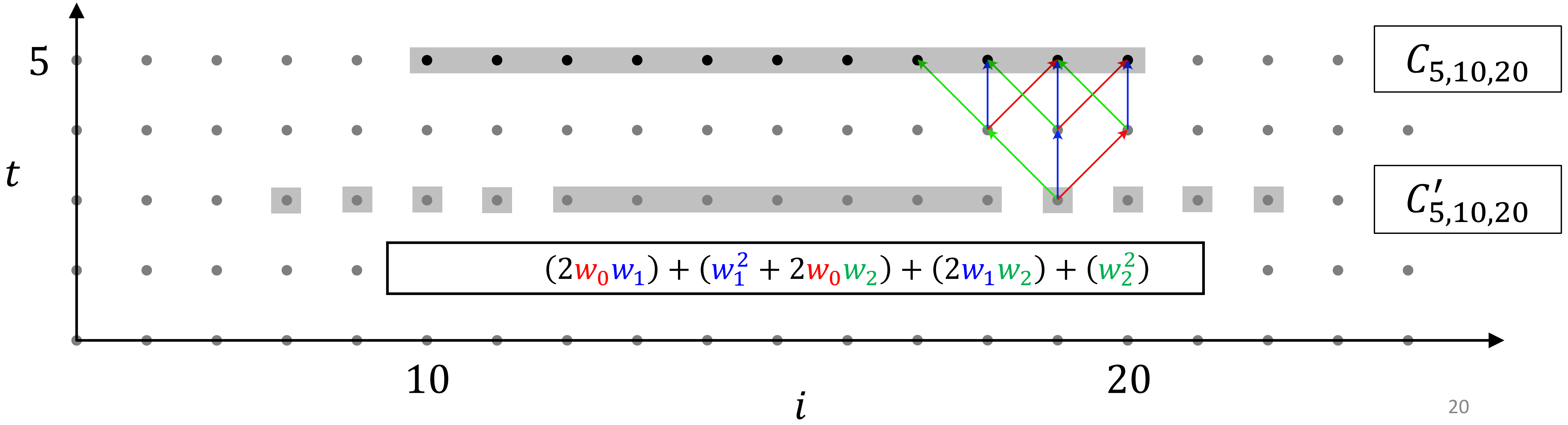
$$C'_{5,10,20} = (\ldots)X_{3,8} + (\ldots)X_{3,9} + (\ldots)X_{3,10} + (\ldots)X_{3,11} + (\ldots)\sum_{i=12}^{18} X_{3,i} + (\ldots)X_{3,19} + (\ldots)X_{3,20} + (\ldots)X_{3,21} + (\ldots)X_{3,22}$$



Construct C' using two substitutions

$$C'_{t,l,m} = \sum_{i=l}^{m} \begin{pmatrix} w_0\left(w_0 X_{t-2,i-2} + w_1 X_{t-2,i-1} + w_2 X_{t-2,i}\right) + \\ w_1\left(w_0 X_{t-2,i-1} + w_1 X_{t-2,i} + w_2 X_{t-2,i+1}\right) \\ w_2\left(w_0 X_{t-2,i} + w_1 X_{t-2,i+1} + w_2 X_{t-2,i+2}\right) \end{pmatrix}$$

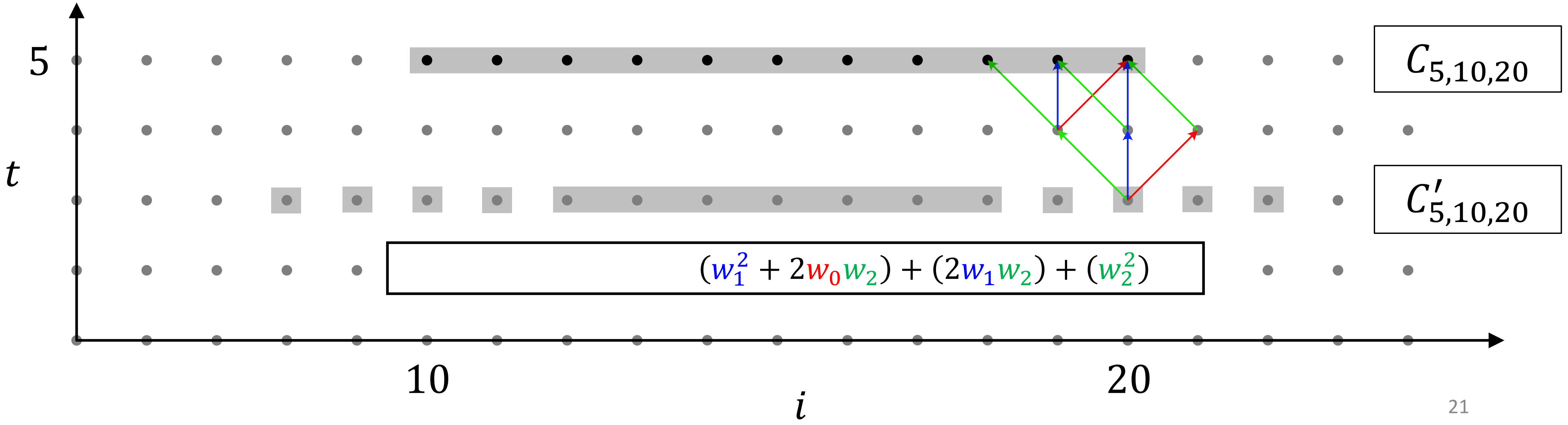
$$C'_{5,10,20} = (\dots)X_{3,8} + (\dots)X_{3,9} + (\dots)X_{3,10} + (\dots)X_{3,11} + (\dots)\sum_{i=12}^{18} X_{3,i} + (\dots)X_{3,19} + (\dots)X_{3,20} + (\dots)X_{3,21} + (\dots)X_{3,22}$$



Construct C' using two substitutions

$$C'_{t,l,m} = \sum_{i=l}^{m} \begin{pmatrix} w_0\left(w_0 X_{t-2,i-2} + w_1 X_{t-2,i-1} + w_2 X_{t-2,i}\right) + \\ w_1\left(w_0 X_{t-2,i-1} + w_1 X_{t-2,i} + w_2 X_{t-2,i+1}\right) \\ w_2\left(w_0 X_{t-2,i} + w_1 X_{t-2,i+1} + w_2 X_{t-2,i+2}\right) \end{pmatrix}$$

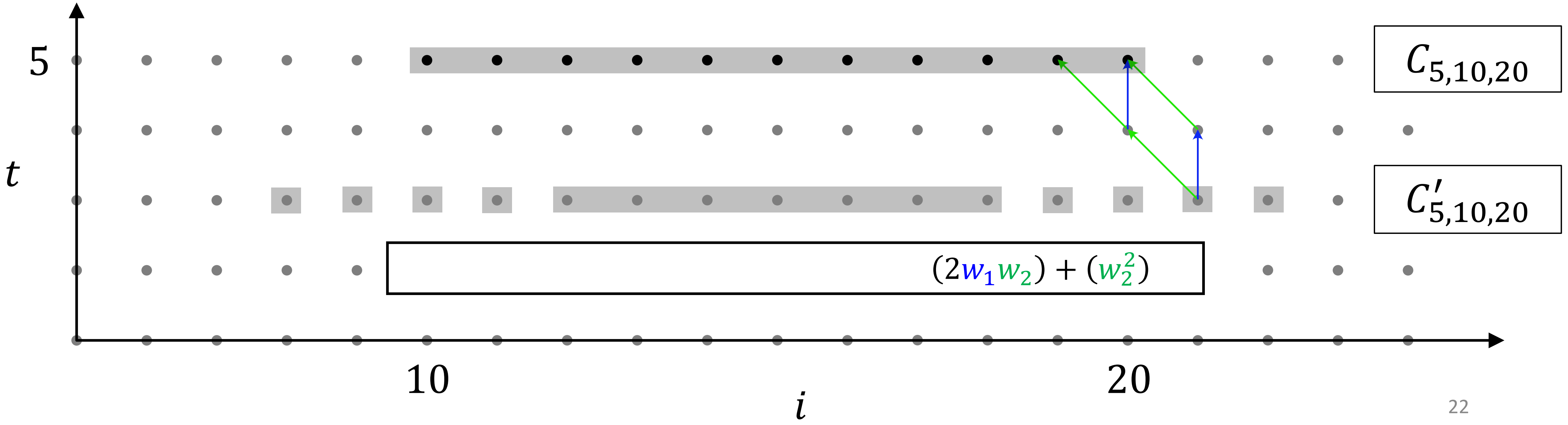
$$C'_{5,10,20} = (\ldots)X_{3,8} + (\ldots)X_{3,9} + (\ldots)X_{3,10} + (\ldots)X_{3,11} + (\ldots)\sum_{i=12}^{18} X_{3,i} + (\ldots)X_{3,19} + (\ldots)X_{3,20} + (\ldots)X_{3,21} + (\ldots)X_{3,22}$$



Construct C' using two substitutions

$$C'_{t,l,m} = \sum_{i=l}^{m} \left(\begin{array}{r} w_0\left(w_0 X_{t-2,i-2} + w_1 X_{t-2,i-1} + w_2 X_{t-2,i}\right) + \\ w_1\left(w_0 X_{t-2,i-1} + w_1 X_{t-2,i} + w_2 X_{t-2,i+1}\right) \\ w_2\left(w_0 X_{t-2,i} + w_1 X_{t-2,i+1} + w_2 X_{t-2,i+2}\right) \end{array} \right)$$

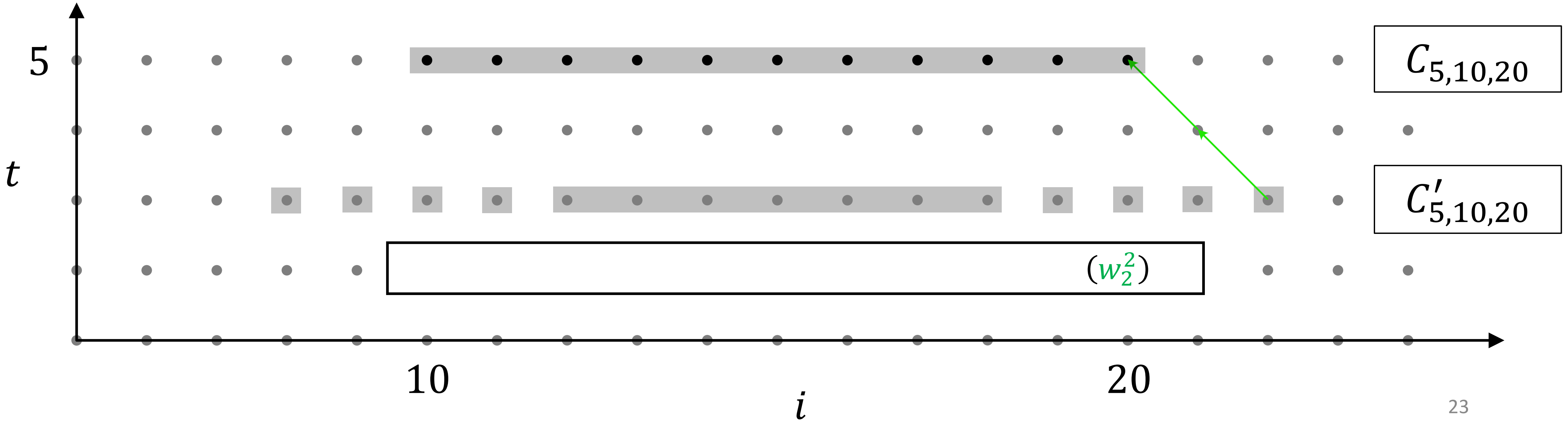
$$C'_{5,10,20} = (\ldots)X_{3,8} + (\ldots)X_{3,9} + (\ldots)X_{3,10} + (\ldots)X_{3,11} + (\ldots)\sum_{i=12}^{18} X_{3,i} + (\ldots)X_{3,19} + (\ldots)X_{3,20} + (\ldots)X_{3,21} + (\ldots)X_{3,22}$$



Construct C' using two substitutions

$$C'_{t,l,m} = \sum_{i=l}^{m} \begin{pmatrix} w_0\left(w_0 X_{t-2,i-2} + w_1 X_{t-2,i-1} + w_2 X_{t-2,i}\right) + \\ w_1\left(w_0 X_{t-2,i-1} + w_1 X_{t-2,i} + w_2 X_{t-2,i+1}\right) \\ w_2\left(w_0 X_{t-2,i} + w_1 X_{t-2,i+1} + w_2 X_{t-2,i+2}\right) \end{pmatrix}$$

$$C'_{5,10,20} = (\dots)X_{3,8} + (\dots)X_{3,9} + (\dots)X_{3,10} + (\dots)X_{3,11} + (\dots)\sum_{i=12}^{18} X_{3,i} + (\dots)X_{3,19} + (\dots)X_{3,20} + (\dots)X_{3,21} + (\dots)X_{3,22}$$

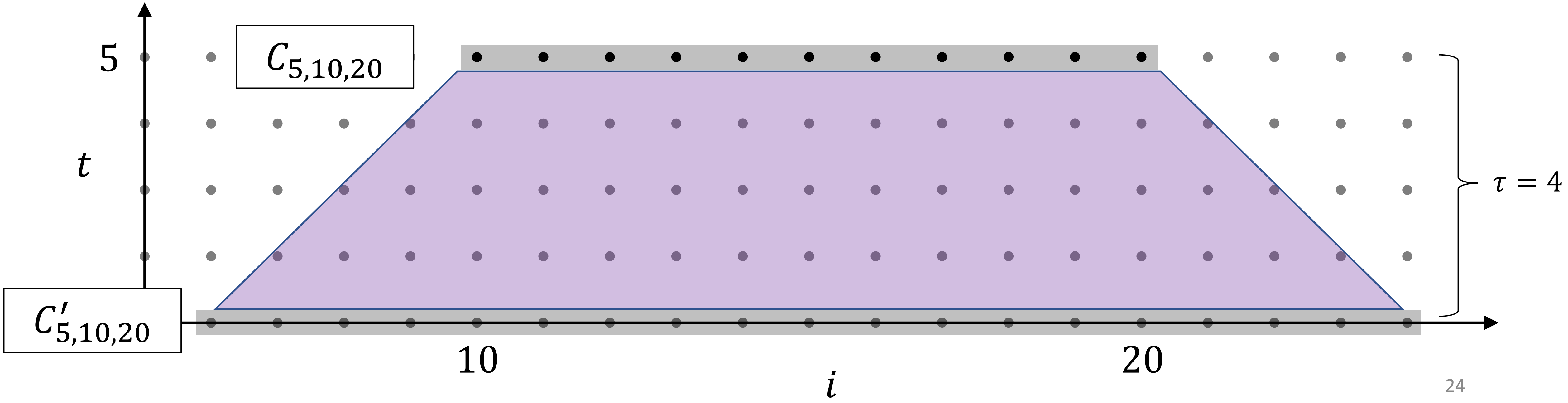


Construct C' using **multiple**, say τ, substitutions

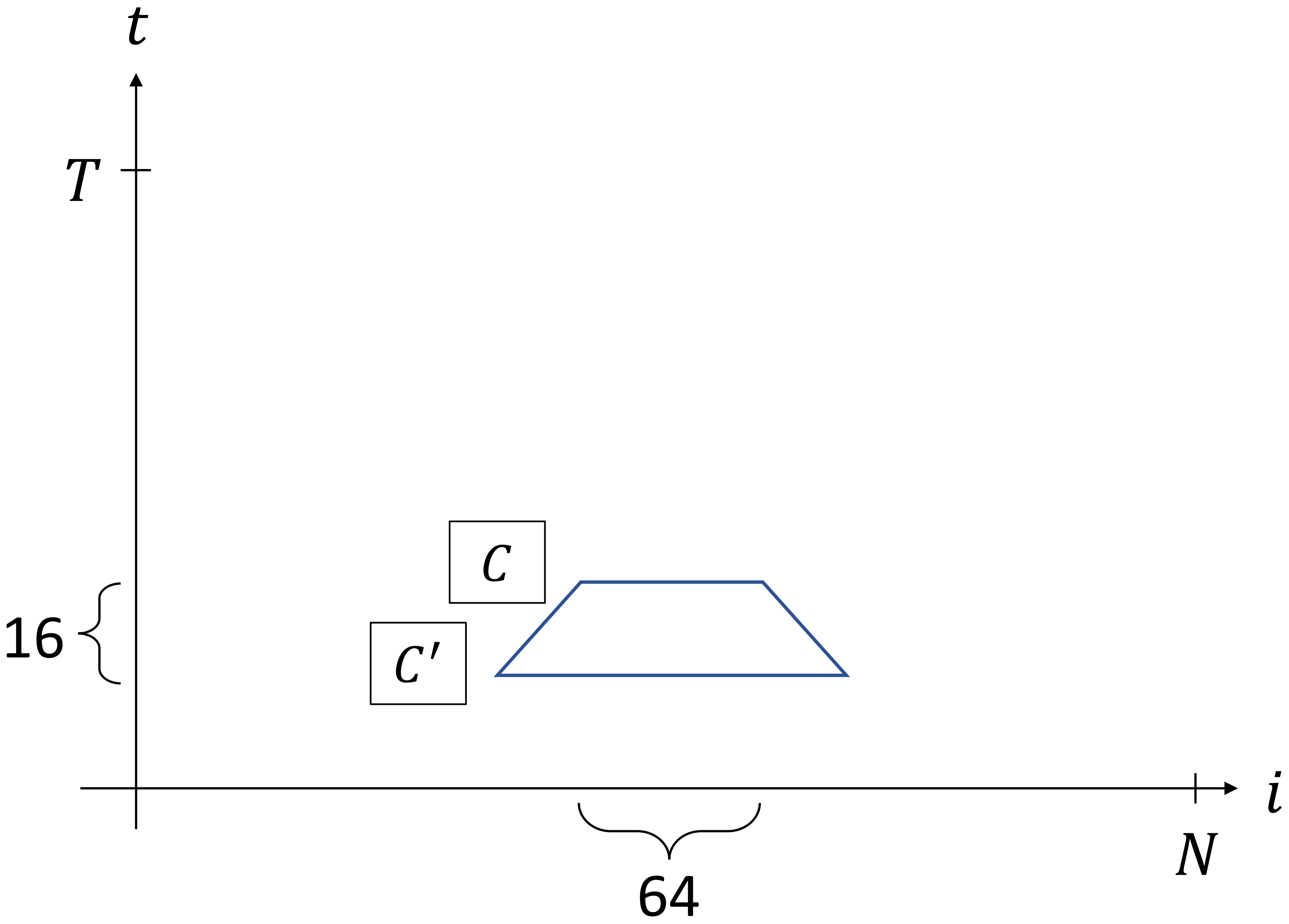
$$\Delta C_{t,l,m} \equiv \left| C_{t,l,m} - C'_{t,l,m} \right|$$

$$\Delta C_{t,l,m} > \text{threshold}$$

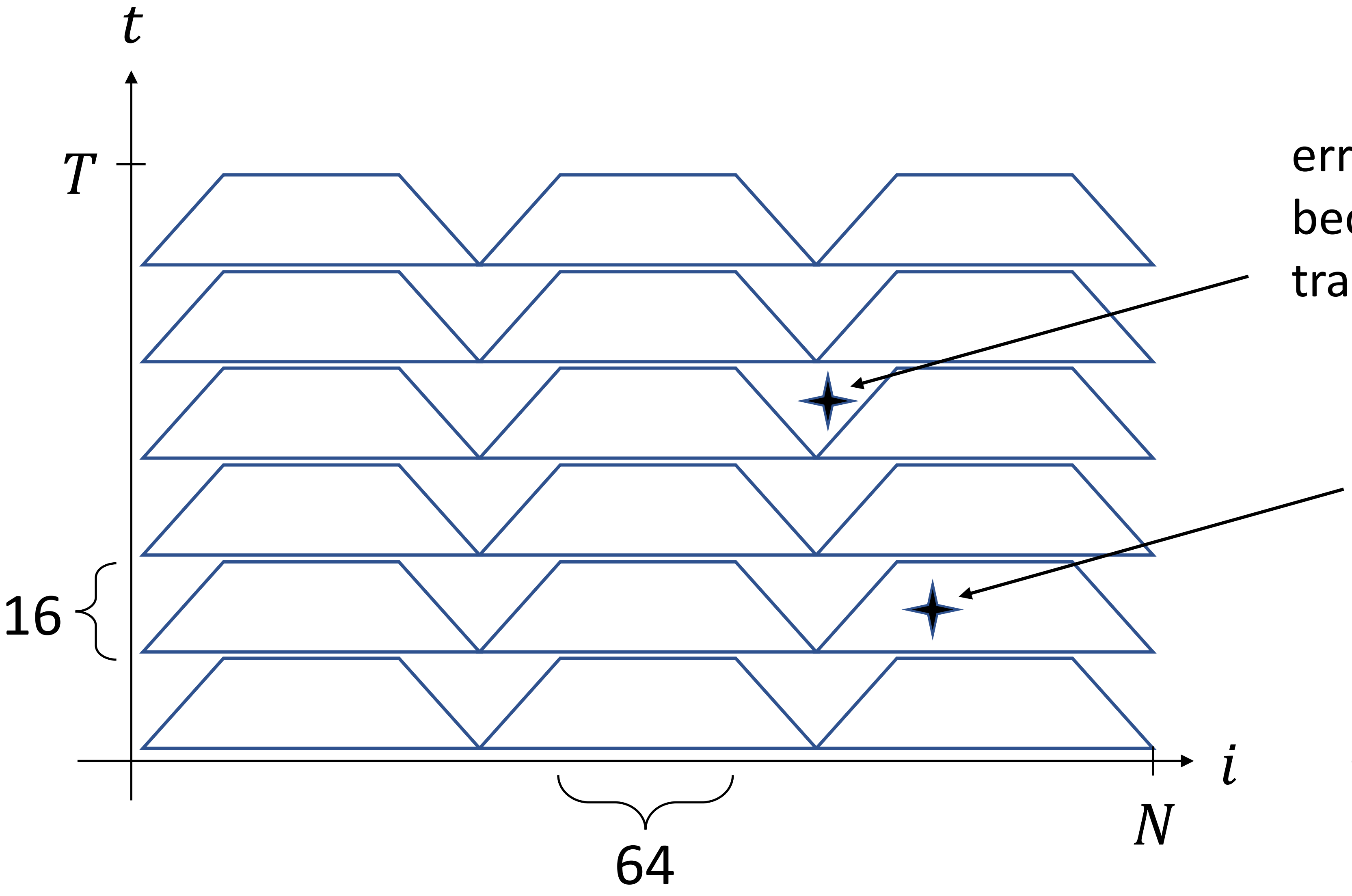
if observed then we have detected an error somewhere inside trapezoidal region



How do we actually use this?



Replicate trapezoidal checksums across the stencil domain



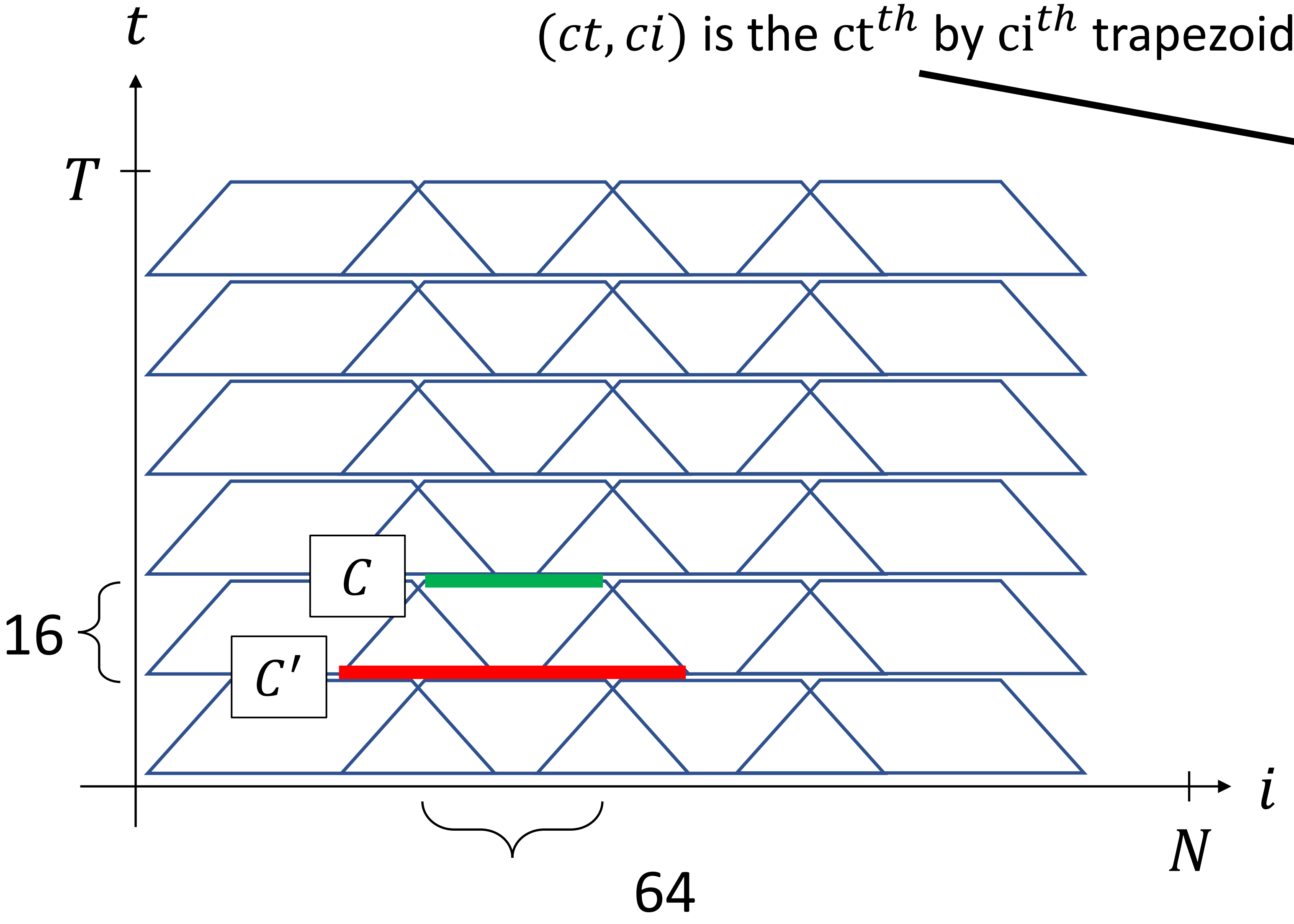
error that can ****not**** be detected because it is not covered by any trapezoid

error that can be detected

*Keep in mind these are ****not***** loop tiles, they are error detection subdomains*

Replicate **and overlap** trapezoidal checksums across the stencil domain

(ct, ci) is the ctth by cith trapezoid

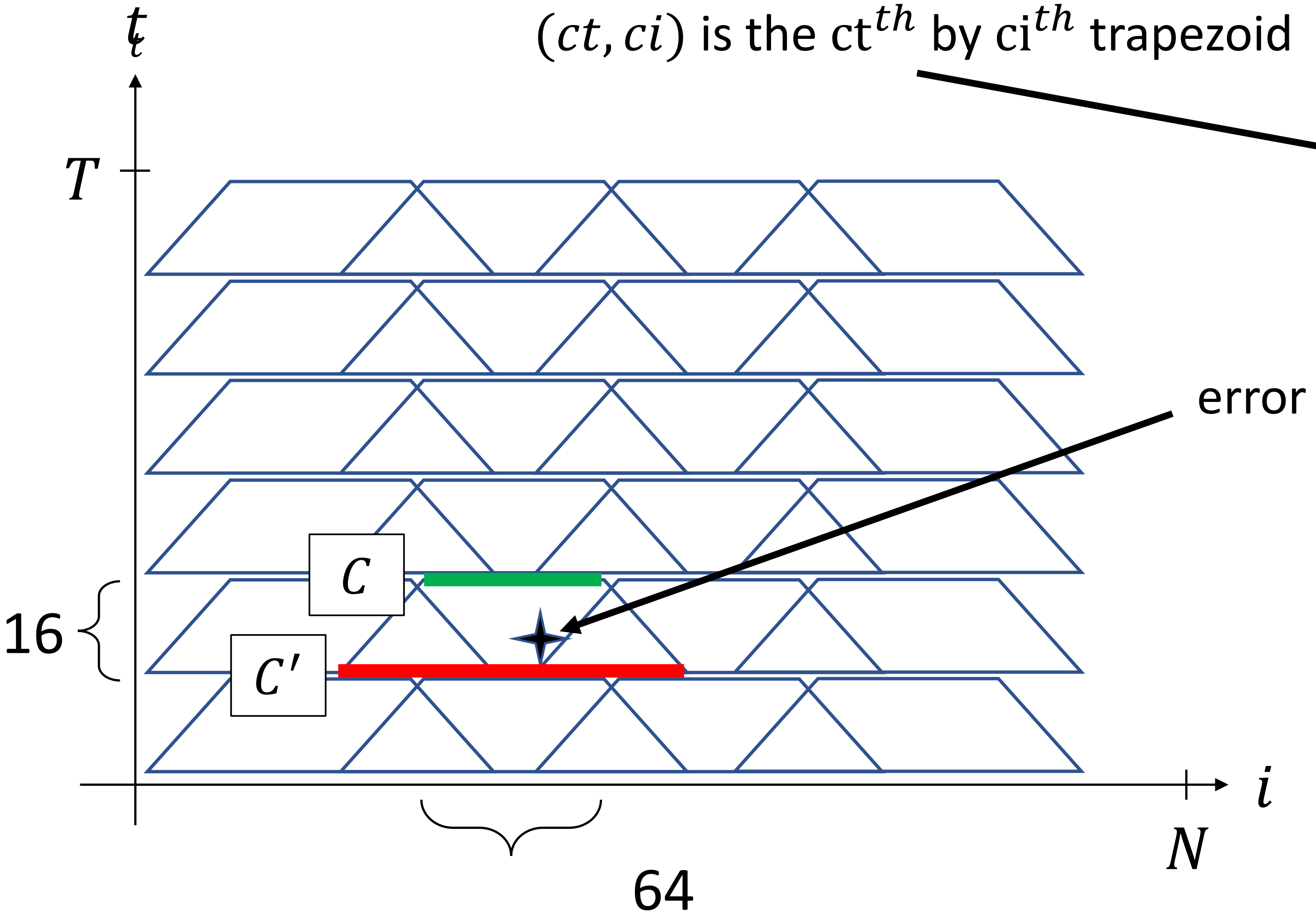



```
$ ./Jac1d1r.check T N
Execution time : 0.000510 sec.
B_NR_checksum_0(1,1)=139264994557.21
B_NR_checksum_0(1,2)=141196794386.83
B_NR_checksum_0(1,3)=140623413185.50
B_NR_checksum_0(2,1)=135273328410.50
B_NR_checksum_0(2,2)=136488421119.67
B_NR_checksum_0(2,3)=136222897074.97
B_NR_checksum_0(3,1)=131309724990.31
B_NR_checksum_0(3,2)=131955997933.86
B_NR_checksum_0(3,3)=131969601885.39
B_NR_checksum_0(4,1)=127395028177.96
B_NR_checksum_0(4,2)=127619756537.45
B_NR_checksum_0(4,3)=127855183219.92
B_NR_checksum_alt_0(1,1)=139264994557.21
B_NR_checksum_alt_0(1,2)=141196794386.83
B_NR_checksum_alt_0(1,3)=140623413185.50
B_NR_checksum_alt_0(2,1)=135273328410.50
B_NR_checksum_alt_0(2,2)=136488421119.67
B_NR_checksum_alt_0(2,3)=136222897074.97
B_NR_checksum_alt_0(3,1)=131309724990.32
B_NR_checksum_alt_0(3,2)=131955997933.86
B_NR_checksum_alt_0(3,3)=131969601885.39
B_NR_checksum_alt_0(4,1)=127395028177.96
B_NR_checksum_alt_0(4,2)=127619756537.45
B_NR_checksum_alt_0(4,3)=127855183219.92
```

C

C'

Errors manifest as sufficiently large differences between C and C'

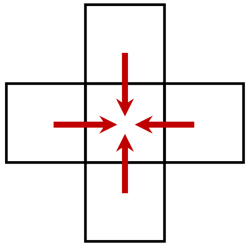



```
$ ./Jac1d1r.check T N
Execution time : 0.000510 sec.
B_NR_checksum_0(1,1)=139264994557.21
B_NR_checksum_0(1,2)=141196794386.83
B_NR_checksum_0(1,3)=140623413185.50
B_NR_checksum_0(2,1)=135273328410.50
B_NR_checksum_0(2,2)=114835478873.24
B_NR_checksum_0(2,3)=136222897074.97
B_NR_checksum_0(3,1)=131309724990.31
B_NR_checksum_0(3,2)=131955997933.86
B_NR_checksum_0(3,3)=131969601885.39
B_NR_checksum_0(4,1)=127395028177.96
B_NR_checksum_0(4,2)=127619756537.45
B_NR_checksum_0(4,3)=127855183219.92
B_NR_checksum_alt_0(1,1)=139264994557.21
B_NR_checksum_alt_0(1,2)=141196794386.83
B_NR_checksum_alt_0(1,3)=140623413185.50
B_NR_checksum_alt_0(2,1)=135273328410.50
B_NR_checksum_alt_0(2,2)=136488421119.67
B_NR_checksum_alt_0(2,3)=136222897074.97
B_NR_checksum_alt_0(3,1)=131309724990.32
B_NR_checksum_alt_0(3,2)=131955997933.86
B_NR_checksum_alt_0(3,3)=131969601885.39
B_NR_checksum_alt_0(4,1)=127395028177.96
B_NR_checksum_alt_0(4,2)=127619756537.45
B_NR_checksum_alt_0(4,3)=127855183219.92
```

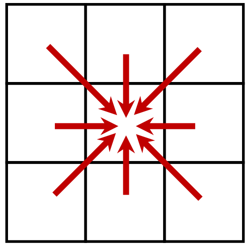
C

C'

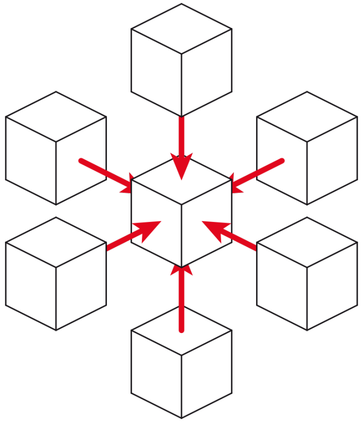
Stencils come in all shapes and sizes, we don't want to do this manually



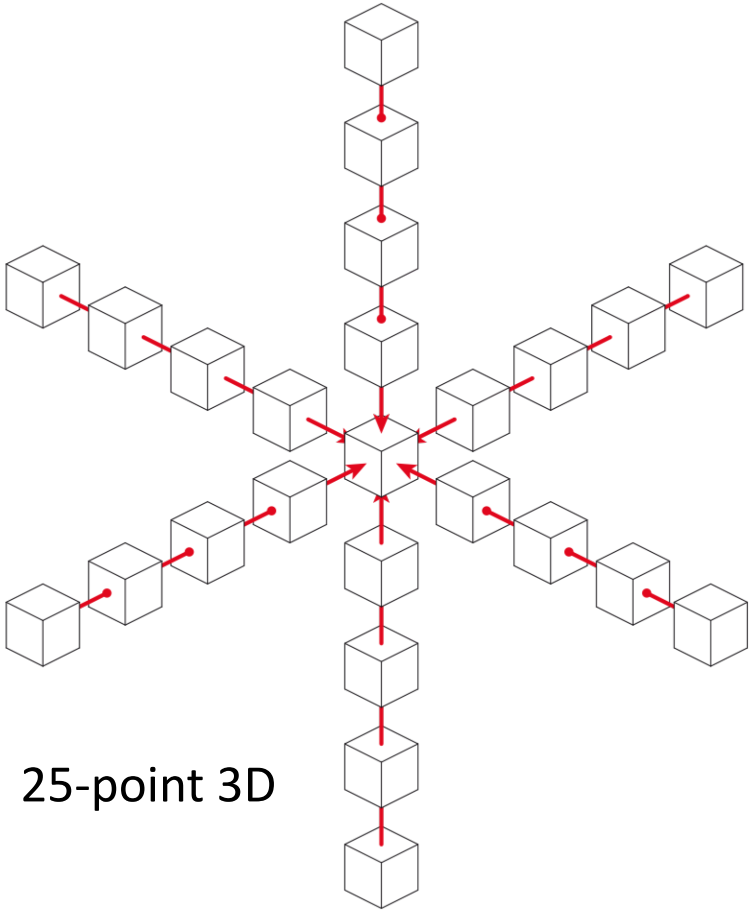
5-point 2D



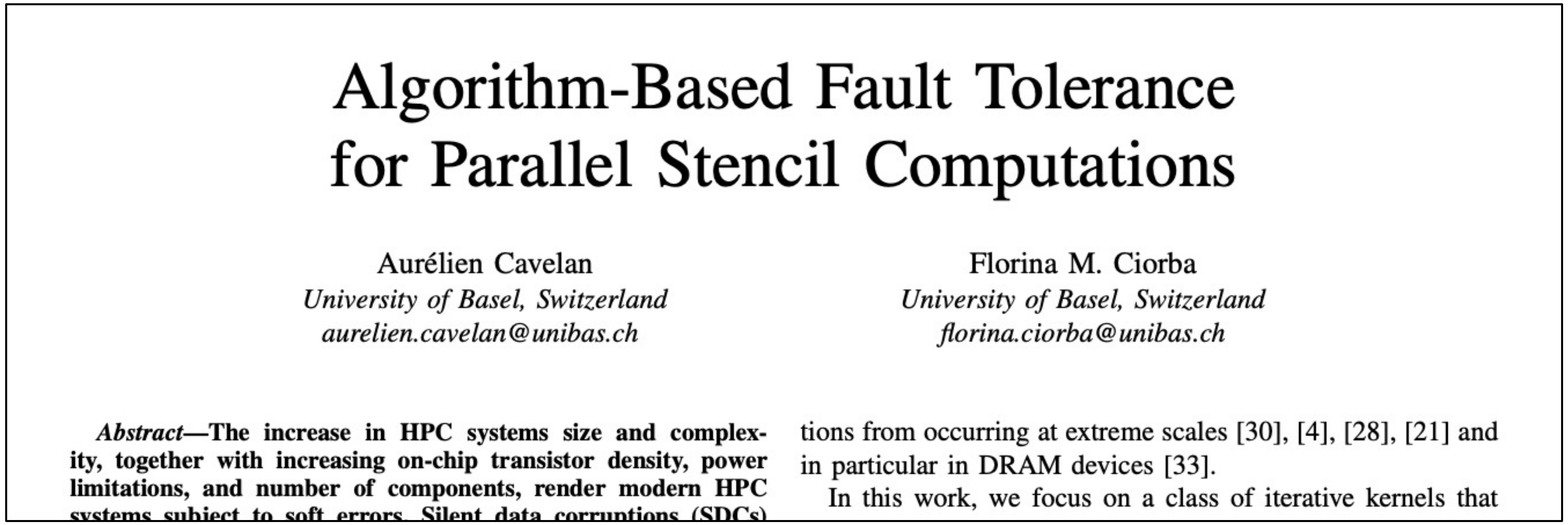
9-point 2D



7-point 3D



25-point 3D



Algorithm-Based Fault Tolerance for Parallel Stencil Computations

Aurélien Cavelan
University of Basel, Switzerland
aurelien.cavelan@unibas.ch

Florina M. Ciorba
University of Basel, Switzerland
florina.ciorba@unibas.ch

***Abstract*—The increase in HPC systems size and complexity, together with increasing on-chip transistor density, power limitations, and number of components, render modern HPC systems subject to soft errors. Silent data corruptions (SDCs)**

tions from occurring at extreme scales [30], [4], [28], [21] and in particular in DRAM devices [33].

In this work, we focus on a class of iterative kernels that

A. Cavelan and F. Ciorba, "Algorithm-Based Fault Tolerance for Parallel Stencil Computations," *2019, IEEE International Conference on Cluster Computing*.

Automatically inferring ABFT checksums for stencils

Alpha

- Domain specific language for manipulating equations
- Models programs as systems of affine recurrence equations (SARE)
- Useful because we need to manipulate algebraic identities

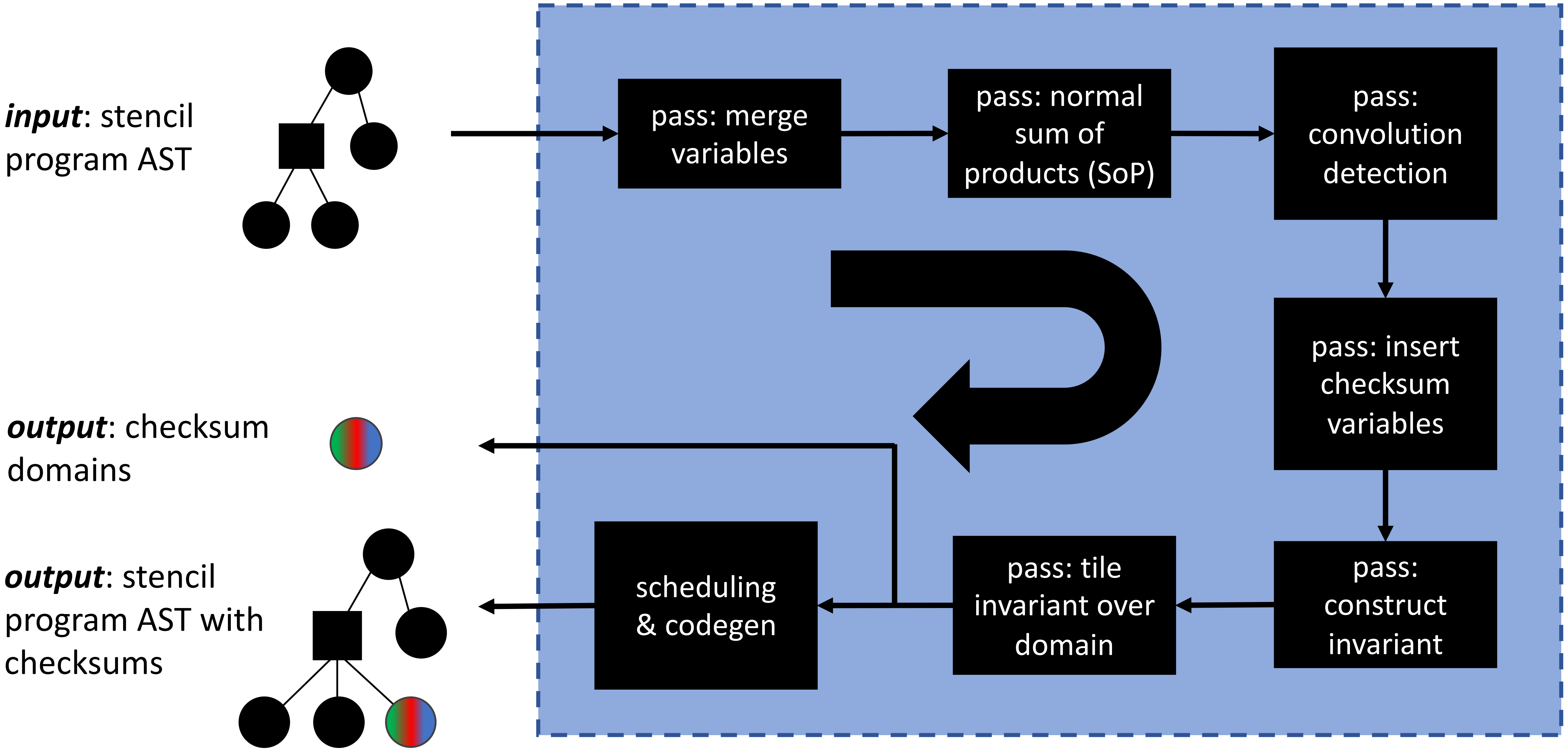
Principally, we carry out ABFT inside Alpha in three steps:

1. Construction of checksum pairs C and C'
2. Replication of checksums over program domain for error coverage
3. Scheduling and code generation

Jacobi 1D 3-point stencil as an Alpha program

```
affine Jac1d1r [T,N]->{ : N>0 and T>0}
inputs
  I : [N+1]              // 1D input
  w0, w1, w2 : []        // scalar weights
outputs
  X: [T+1,N+1]           // 2D output
let
  X[t,i] = case {
    {: t=0 }             : I[i];
    {: t>0 and i=0}      : X[t-1,0];
    {: t>0 and 0<i<N}    : w0 * X[t-1,i-1] + \
                           w1 * X[t-1,i] + \
                           w2 * X[t-1,i+1];
    {: t>0 and i=N}      : X[t-1,N];
  };
.
```

Automatic ABFT analysis inside Alpha



Merge coupled variables into single variable

Input: list of equations with shared dependencies variables

```
E[t,i] = case {
  {: t>0 and i=0} : E[t-1,0]
  {: t>0 and 0<i<=N} : E[t-1,i] - a[i]*(H[t-1,i] - H[t-1,i-1])
}

H[t,i] = case {
  {: t>0 and 0<=i<N} : H[t-1,i] - b[i]*(E[t-1,i+1] - E[t-1,i])
  {: t>0 and i=N} : H[t-1,N]
}
```

Output: single equation

```
M[t,i,z] = case {
  {: t>0 and i=0 and z=0} : M[t-1, 0,z]
  {: t>0 and 0<i<=N and z=0} : M[t-1,i,z] - a[i]*(M[t-1,i,z+1] - M[t-1,i-1,z+1])
  {: t>0 and 0<=i<N and z=1} : M[t-1,i,z] - b[i]*(M[t-1,i+1,z-1] - M[t-1,i,z-1])
  {: t>0 and i=N and z=1} : M[t-1,N,z]
}
```

Normalize sum of products (SoP) expressions

Input: addition/subtraction expression where each term involves stencil variable

```
X[t-1,i] - a[i]*(X[t-1,i] – X[t-1,i-1])
```

Output: addition expression where terms are of the form BIN_OP(weights, stencil_var)

```
(1)*X[t-1,i] + (-1*a[i])*X[t-1,i] + (a[i])*X[t-1,i-1])
```

Trade off between error coverage and overhead

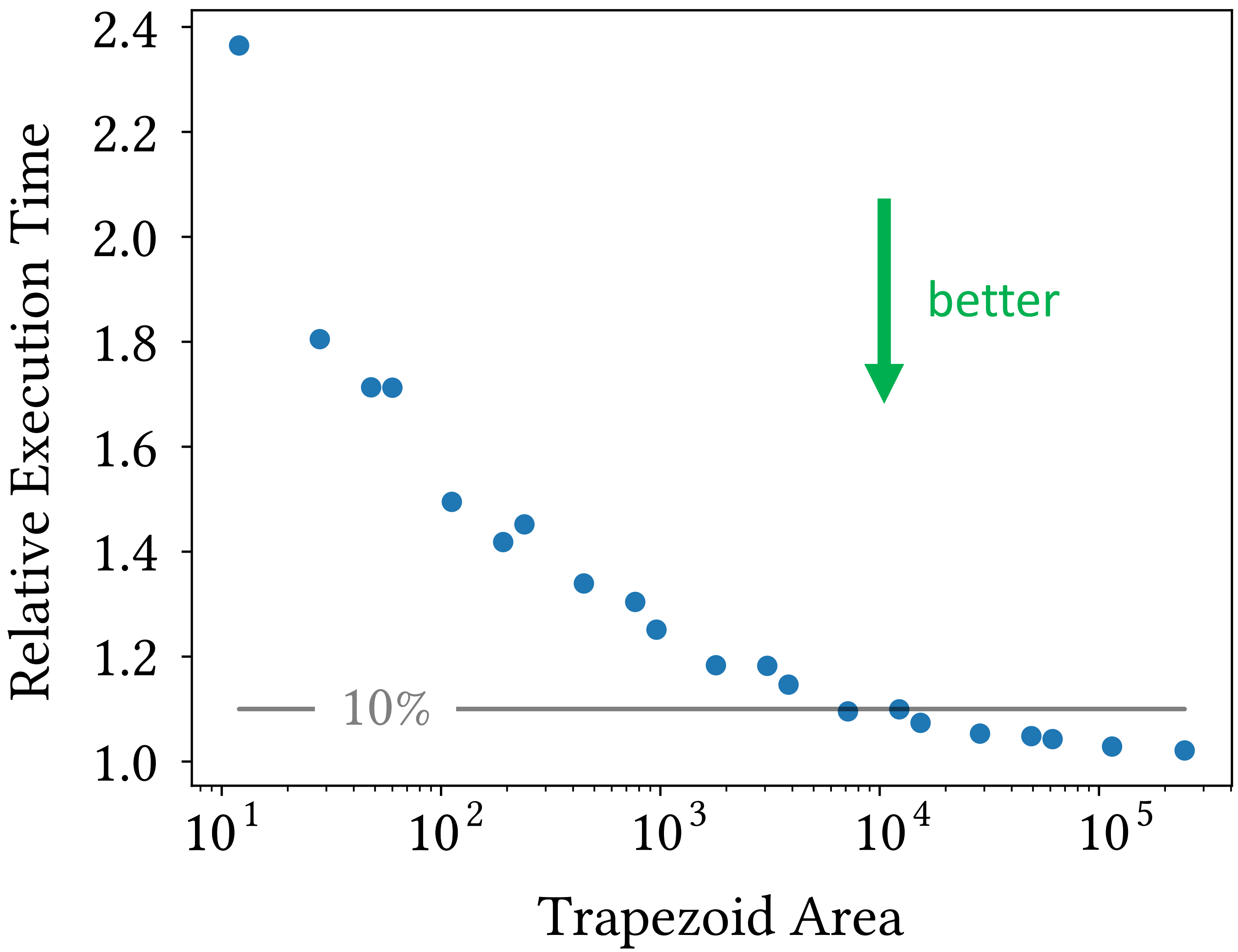
Smaller trapezoidal checksum regions

- Higher coverage
- Require more work (compute checksums more frequently)

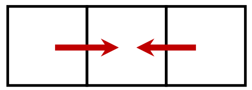
Larger trapezoidal checksum regions

- Lower coverage (errors likely to be swallowed)
- Require less work

Overhead decreases with trapezoidal checksum size

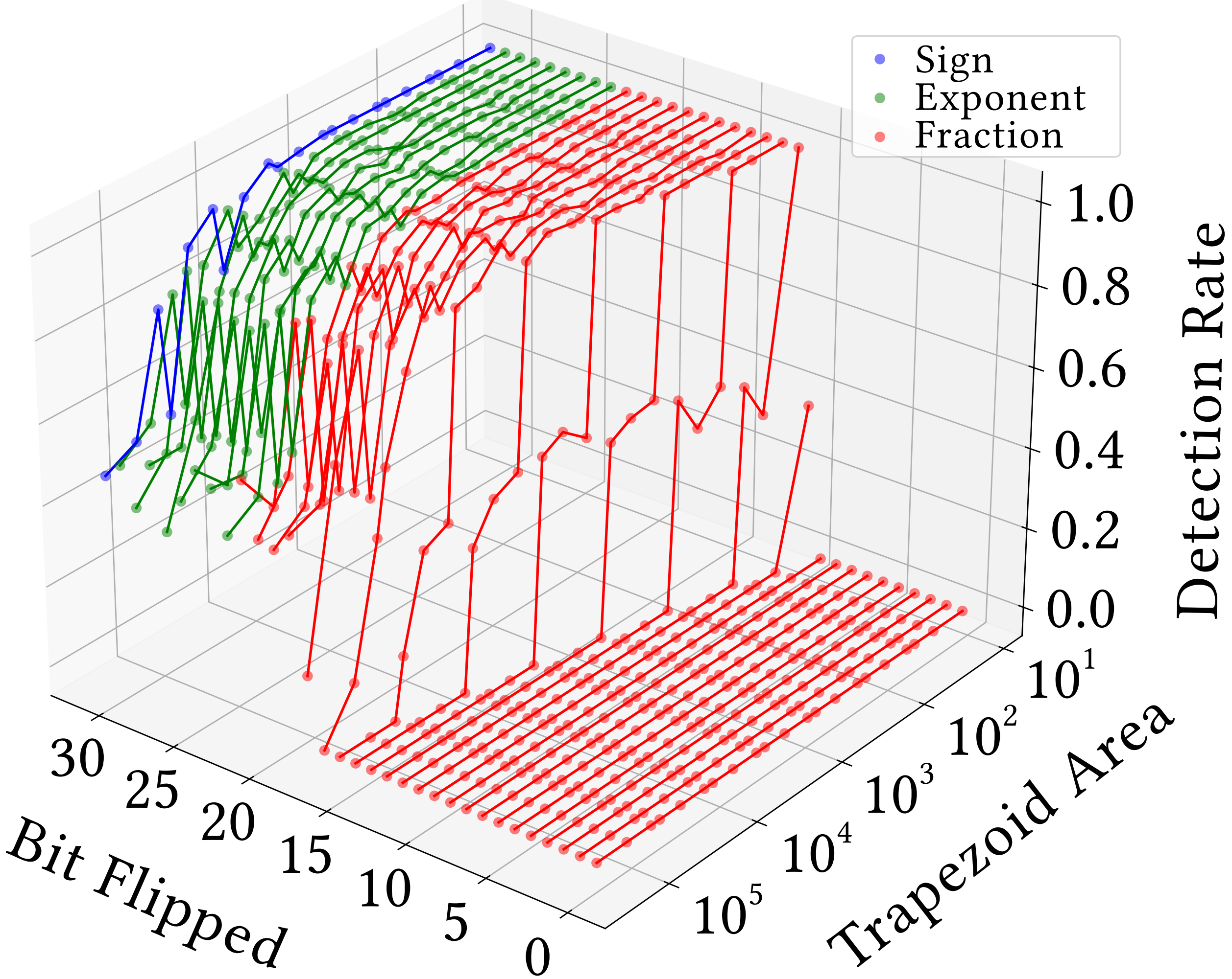


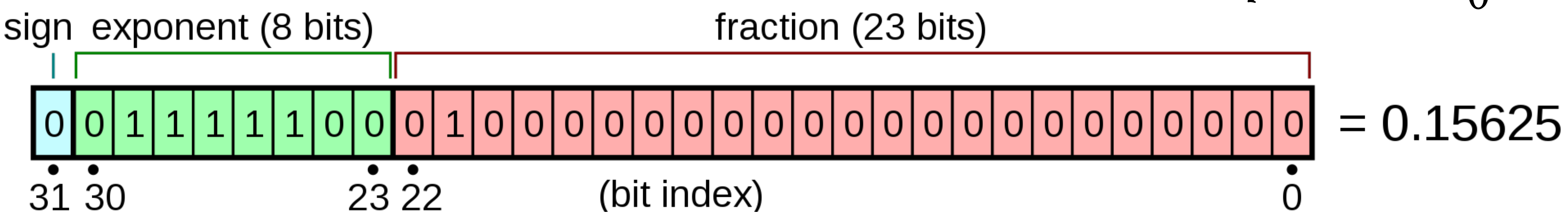
3-point 1D stencil



Detection rate decreases with trapezoidal checksum size

- Single precision (32 bit) floating point
- Each point is the fraction of 100 trials where ΔC is above the detection threshold for the given bit-flip and trapezoid-area pair





Open questions and future work

We have shown:

- Application-specific technique for silent error detection based on algebraic properties
- Illustrated how it can be automated

What's next?

- Experimental evaluation on more "realistic" stencils
- Trade-off between tolerance and detection effectiveness
- How general is this, really?

thanks