January 25, 2009

1 Model Computing

The basic operation in COHO is to compute the forward reachable space at each step. In COHO, we use projectagon to approximate and represent reachable space.

Therefore, a straight-forward method to move forward a projectagon is

- Find all faces of the projectagon.
- Compute the linearized model within a region which contains all reachable space for each face.
- Compute a timestep which make sure the model is valid.
- Move forward the face.
- Project down onto all slices of the projectagon.

However, projecting all forward faces on to all 2d bases is time consuming, and the result is too conservative because the model has large linearization error on some dimensions. Our solution is that

- For each slice, find all faces corresponds to its polygon segments.
- Compute the model and time step, and move forward the face.
- Project down onto the current slices *ONLY*.

Obviously, the method is more efficient. The error is smaller because the forward face is projected onto slices with large error.

1.1 Computation of time step

The time step can be computed by finding the maximum derivative over the model region. However, the time step computed by this method is usually too small. To increase the step size, we use the *assume-guarantee* method

- Assume the maximum moving distance on each direction for all points on the faces, compute the corresponding model.
- Guess a time step and try to move forward the face.
- Verify that the forward face is within the maximum distance.

1.2 How to find the face of projectagon

For each segment of a slice, the corresponding face must be contained by the intersection of segment and the convex hull of space. However, the space is usually non-convex. Therefore, we developed *interval closure* method to reduce the error

- Given a segment of a slice, the segment endpoints gives the range of slice base variables.
- The range is applied to other slices, range of other variables is computed based on the non-convex polygons.
- A bounding box is returned until the range of all variables converges to fixed values.

The intersection of segment and the bounding box is a more tight approximation of the face.

1.3 The height of face to move forward

In the method presented above, the result is under-approximated if we move forward faces corresponds to slice segment only. The points from other faces may pass over the face and reach further place, as shown in figure 1. Our

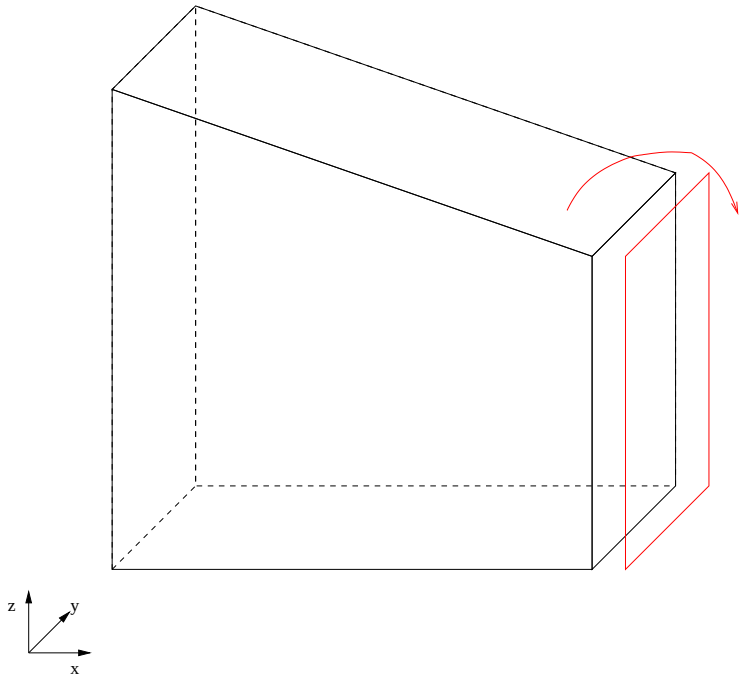

Figure 1: Possible under approximated result

solution is to increase the height of faces to block all trajectories from interior faces.

The method is (as shown in figure 2)

- Assume the maximum moving distance allowed (bloatAmt) on direction x is x_+ and x_- over all points on the boundary of space.
- Bloat segment inward by $x_- + x_+$, compute the bounding box using interval closure method.
- The face is the intersection of segment and the bounding box.
- Increase the height of face by $x_- + x_+$ to make sure the result is over-approximated.
- Compute the model and move forward the face.
- Check the maximum bloatAmt for the forward face.

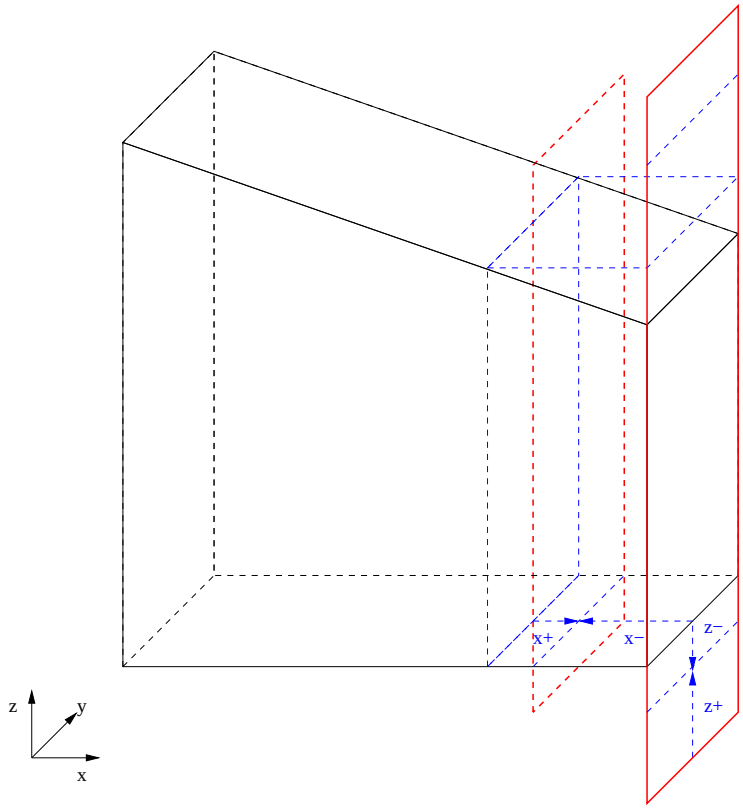

Figure 2: The height of face to move forward

The soundness of the method is "proved". Any points can not pass over the red-dotted face by the assumption. However, the proof has a problem: x_- and x_+ is the maximized moving distance for all faces corresponds to projection polygon. The moving distance for points on interior face which does not corresponds to polygon segment of this slice is not bounded by x_- and x_+, and it is not verified at the final step. Therefore, it is still possible that some trajectories from interior points may over pass the red-dotted face.

An alternative method is to move forward the space between the face and the blue dotted face (show in the figure), not only the face. But we have similar problem to prove its correctness.

1.4 Iterative method to reduce model error

The same bloatAmt and time step are applied for all faces for correctness. However, a specified face usually has smaller bloatAmt than the global one. Increasing the height of all faces by $x_- + x_+$ is also usually too conservative.

Therefore, we use an iterative method to reduce error

- Given a bloatAmt, compute the model.
- Compute the maximum derivative over the model.
- The maximum moving distance is timeStep*(maximum derivative).
- Set bloatAmt as the maximum moving distance compute at the above step and repeat.
- Exit until the bloatAmt is accurate enough and move forward the face.

1.5 Using different bloatAmt for each face

Another method to reduce model error is to use different bloatAmt for each face. It is not difficult to guess the global bloatAmt using the bloatAmt from previous steps, it is non-trivial to guess a bloatAmt for every face.

Our solution is that

- Move forward a projectagon using global bloatAmt as the usual way.
- Compute the *realBloatAmt* for each face from the result
- Use the *realBloatAmt* as the maximum moving distance for each face, compute the model and move forward the projectagon again.

This method is optimal. It will increase running time but decrease the model error.

2 Changing Variable

In the arbiter circuit, the interval for internal nodes $i1$ and $i2$ is too large to verify the circuit. However, $i1$ and $i2$ should be close to its equilibrium point. Therefore, we change variables to solve the problem.

$$\begin{aligned} q(r1, x1, x2) &= i1 | ids_n(gnd, r1, i1) = ids_n(i1, x2, x1) \\ u &= i - q \end{aligned}$$

In the new system, the derivative is computed as

$$
\begin{aligned}
\frac{dx1}{dt} &= -(ids_n(i1, x2, x1) + ids_p(vdd, r1, x1) + ids_p(vdd, x2, x1))/C(x1) \\
&\quad -(ids_n(q(r1, x1, x2) + u1, x2, x1) + ids_p(vdd, r1, x1) + ids_p(vdd, x2, x1))/C(x1) \\
\frac{di1}{dt} &= (ids_n(i1, x2, x1) - ids_n(gnd, r1, i1))/C(i1) \\
\frac{du1}{dt} &= \frac{di1}{dt} - \frac{\partial q}{\partial r1} \cdot \frac{dr1}{dt} - \frac{\partial q}{\partial x2} \cdot \frac{dx2}{dt} - \frac{\partial q}{\partial x1} \cdot \frac{dx1}{dt}
\end{aligned}
$$

Using the new variable, we have more constraints as

$$
\begin{aligned}
u &\in [-1.45, 1.45] \\
u &\in [-q, 1.45 - q] \\
u1 &<= x1
\end{aligned}
$$