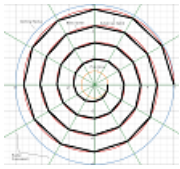
Modeling and Orb-Weaver Capture Spiral

Benjamin Smith
Department of Computer Science
Carnegie Mellon University
bs1@andrew.cmu.edu



Abstract

Spider webs have been the subject of numerous models throughout the years, however the inward spiral of sticky thread known as the capture spiral has traditionally been neglected in such endeavors. This paper seeks to model this structure as an inward spiral of straight lines connecting the more commonly modeled radial threads, while ensuring that the spiral is tight enough to effectively ensnare prey in the web. The ultimate model produced is unable to replicate this form in its entirety, however a pair of models were produced by which the full structure can be produced. A basic spiral structure generates anchor points where the capture spiral connects to the radial framework, while a second model uses those points to individually construct each straight line segment of the full capture spiral. Both ensure the spiral has no gaps large enough for prey of a preset size to slip through.

1. Introduction

Orb-weaver spiders (spiders of the family Araneidae) are well-known for the intricate silk webs they weave in order to catch prey. These webs generally consist of four components: a triangle of anchor threads and frame threads spread between a set of anchor points, a framework of radial threads extending from the center to those anchor points, a secondary frame of threads in concentric circles strung between the radial framework, and, finally, the capture spiral. It is this last component, constructed after all the others, that this paper will be focusing on. The capture spiral is an inward spiral from the edge of the anchor threads towards the center of the web, strung up between the radial framework threads using the secondary frame as scaffolding. It is created with the adhesive threads that are used to capture and restrain prey, rather than the more sturdy construction threads that comprise the rest of the web. If this spiral is too loose and the spaces between threads too wide, then prey could pass directly through the web without being caught. If it's too tight then the spider is wasting resources making more thread than is

necessary, which is highly detrimental given how often webs need to be repaired (after every capture) or rebuilt completely (often every day).

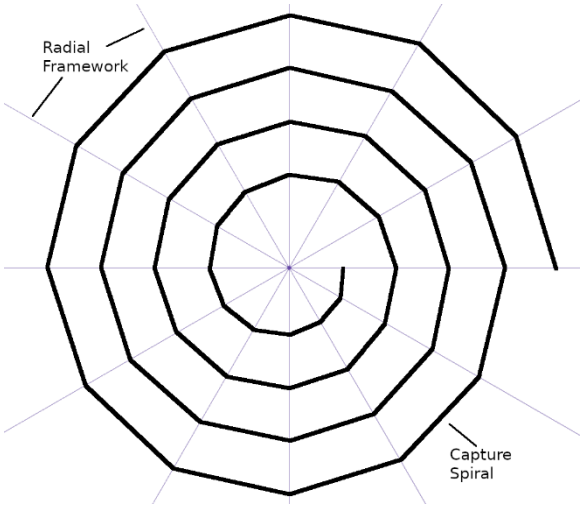


Fig 1: Relevant Parts of an Orb-Weaver Web

For the purposes of this paper, all aspects of the web beyond the capture spiral and radial threads will be ignored as they are not directly related to the capture spiral's form. Expanding these models to generate the entire web from start to finish is an enticing avenue of further research, but is well beyond the scope of this paper.

Thus, the desired properties of a model are as such: on a basic level, it must be able to model a spiral from any given starting radius into the center, and it should guarantee that the space between layers of that spiral does not exceed a certain limit. It should also ensure an upper limit on the number of rotations for the sake of efficiency. The solution for these requirements in this model is to divide the web space into a series of rings, each half the size of the upper limit. Each rotation of the spiral must start at the edge one ring and end at the edge of the next ring in, never leaving the ring at any point in between. This guarantees an upper bound on the number of rotations and ensures that the space between one rotation and the next rotation out never exceeds twice the ring width. The modeling will be event-triggered due to the slow and methodical nature of web construction. Beyond the basic level, however, it must construct this spiral out of straight line segments which only alter their direction at pre-set angles relative to the origin.

This provides an interesting challenge, as the spiraling line segments require modeling linear motion which still adheres strictly to a set circular pattern without the use of trigonometric functions. Additionally, the distance constraints for where the spiral can go are set not by external obstacles or individual points, but by the spiral's own continuous, already-modeled path. Having to model movement in such a polar/Cartesian midpoint while accounting for path restrictions also existing in that difficult-to-model half-space would prove quite the challenge to tackle head-on.

As such, some approximations were required. The capture spiral is first approximated as a smooth, continuous spiral in polar coordinates. This is used to find intersection points with the

radial framework that serve as the start and end points of the line segments, which are then constructed individually in cartesian coordinates.

2. Approach

The modeling process was divided into two main steps. First, the generation of a main, basic spiral, and second, creating straight lines between points generated by that spiral to create the final, advanced spiral. The slow, methodical nature of web-spinning lends itself to an event-triggered model rather than a time-triggered one.

2.1. Basic Spiral and Polar Coordinates

For terminology, a cycle of a spiral refers to a section which starts and ends at the same Theta in polar coordinates. Adjacent cycles can have a straight line drawn between them without crossing the spiral at any point.

The modeling of the basic spiral is done in polar coordinates. It begins with a Starting Radius, which is the initial state of the Basic Spiral's Radius variable (Rad), and the radius of the Free Area which the spiral is not allowed to enter (FreeRad). Basic starting requirements are that FreeRad >= 0 and Rad >= FreeRad.

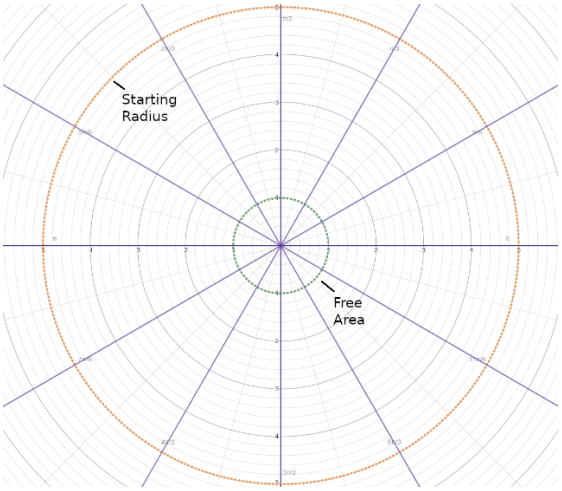


Fig 2: Starting Radius and Free Area

The model will cease running once it reaches the outer edge of the Free Area, as determined by when Rad = FreeRad.

From here the safety structure is created. The size of prey to be captured by the web is represented by pSize > 0. If there are gaps in the spiral larger than pSize, Free Area excluded, then the web will not be serving its purpose to capture the designated prey. Thus it is imperative that cycles of the basic spiral are sufficiently close together.

To achieve this safety, the web is divided into Rings. The distance between the inner and outer edge of each Ring is pSize/2, such that no two points with the same Theta that are in adjacent rings can possibly have a distance between them larger than pSize. The rings start at the Starting Radius and work their way in to the Free Area.

The current Ring is kept track of by the RingRad variable, which measures the distance from the center of the web to the inner edge of the currently active Ring. RingRad starts at Rad-pSize/2.

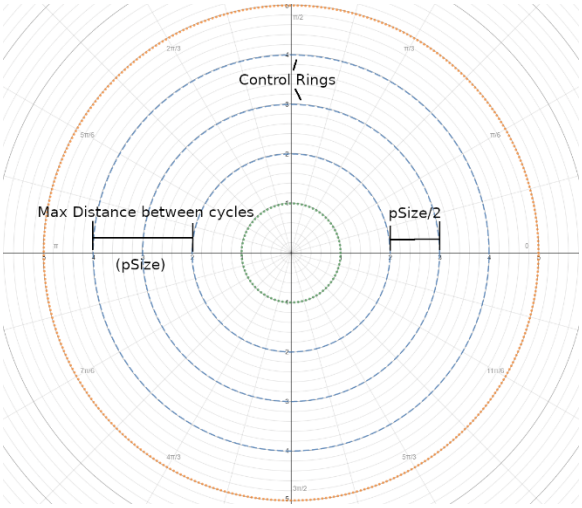


Fig 3: Control Rings

Now the modeling of the Basic Capture Spiral (Basic Spiral) can begin. The current location on the spiral is represented by Rad (which begins at the Starting Radius and is the point's radius in polar coordinates) and Theta (which begins at zero and is the point's angle in polar coordinates). Theta is measured in radians. Theta starting at zero is primarily for simplicity, as the model's circular nature means that once complete it can be rotated to align with any starting angle required.

To ensure that the spiral has no gaps larger than pSize, each cycle of the spiral which starts at Theta = 0 must begin at the outer edge of one of the Rings and end at the inner edge of that Ring. The inner edge of a ring is the outer edge of the Ring adjacent to the interior, thus the spiral is in proper position to begin the next cycle. If adjacent cycles are fully contained within adjacent Rings, then as already described it's not possible for there to be a gap between them larger than pSize.

The spiral is modeled with an Ordinary Differential Equation (ODE) which reduces Rad while progressing Theta, pulling the spiral towards the center of the web as it rotates around said center.

The ODE is event-triggered, and thus can be forced to halt when specific requirements are met rather than simply stopping after a random amount of time. The event that will halt the modeling of the spiral is when Theta meets or exceeds 2*π. This indicates that it has completed a

full cycle since starting the current cycle at 0. This means that the model can re-assert control at this moment and use that control to reset Theta to 0, the radian measurement equivalent of $2*\pi$.

Also at this time, because a full cycle has been completed in the current Ring, we move to the next ring in by subtracting pSize / 2 from RingRad. Rad being equal to RingRad before the subtraction, and each Ring having a width of pSize/2, means that Rad is now on the outer edge of the new active Ring.

If Theta were allowed to start at some arbitrary value Theta_1 instead of 0, then there would be need to two events. The ODE would need to be stopped when Theta = $2*\pi$ to reset it to 0, and it would also need to be stopped when Theta = Theta_1 to advance to the next ring in. Ensuring Theta starts at 0 allows those two events to be combined into one.

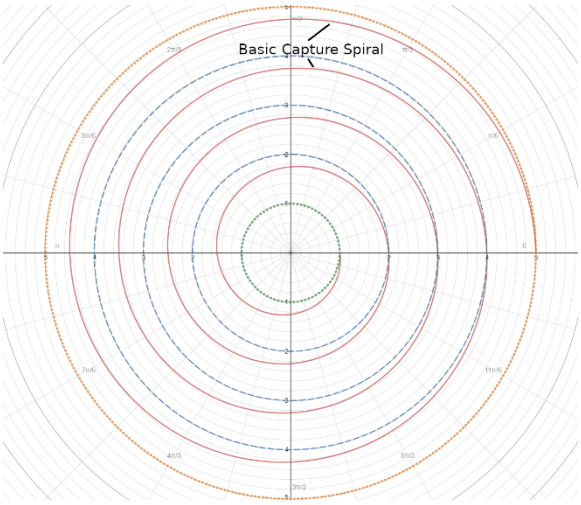


Fig 4: Basic Spiral

Events being dependent on Theta means that Theta sets the pace for how quickly variables progress in the ODE. Theta' is set to 2π so that each cycle of the spiral will be completed in one unit of simulated time. Accordingly, Rad' is set to pSize/2 so that in that time the spiral will traverse its current Ring and reach the boarder of the next ring in just as Theta hits 2π.

Traversing one ring per cycle also ensures efficiency. The rings being width pSize/2 when the requirements only mandate cycles being with distance pSize of each other means that this model is not as efficient as it could be. However, it also ensures that our model has no more than twice the necessary number of cycles. It's not a foolproof efficiency claim, as it's hypothetically possible that the radius fluctuates wildly within a ring and inflates the total length of the Capture Spiral, but it is reasonable insurance. This is especially true given that Rad is never manually set during the model and for every ODE it is a part of, Rad' is strictly negative.

The model is not yet complete. The Basic Spiral has been fully modeled, but it is ultimately only a means to the end that is the Advanced Spiral.

To create the Advanced Spiral, we must subdivide the current model using the web structure's Radial Threads. The number of Radial Threads for a given web is represented by

Radii, and they divide the web into Radii equal sections. Where the Basic Spiral intersects with these Radial Threads will be an anchor points of the Advanced Spiral, thus it is imperative that the ODE be stopped at each intersection so the coordinates can be noted. The model requires Radii >= 4, even though a web could hypothetically be spun between only three Radial Threads. The reason for this will become apparent in the second main step when the Advanced Spiral is generated.

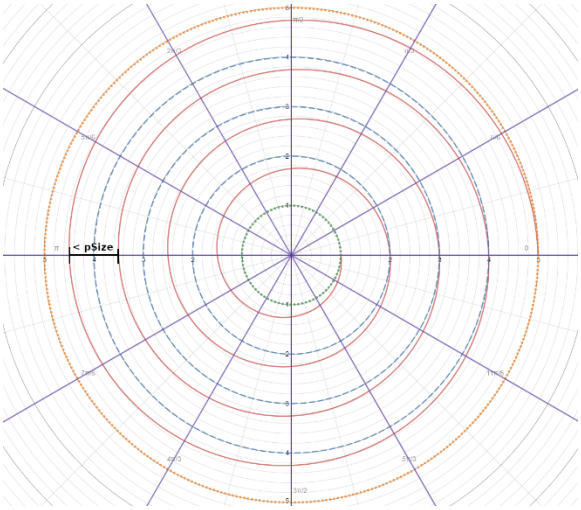


Fig 5: Subdivided Basic Spiral

A secondary theta value for the spiral, TickTheta, keeps track of the current relative position of the spiral, with the angle of the most recently intersected Radial Thread being treated as 0. TickTheta' = 2*π, meaning that it advances at the same speed in the ODE as Theta. The ODE is halted whenever TickTheta = 2*π/Radii. During control in between ODE runs, if TickTheta >= 2* π/Radii it gets reset to 0 in the same way that Theta gets reset if Theta >= 2* π.

Since the Radial Threads divide the web evenly, there is an angle of 2*π/Radii between any two adjacent Radial Threads. Thus because TickTheta gets reset every 2*π/Radii, when it reaches that value again it indicates that another Radial Thread has been reached. This requires the assumption that the Basic Spiral begins on a Radial Thread, which is a reasonable assumption to make because the Capture Spiral must be started by attaching a thread to an anchor point on a Radial Thread.

The addition of the Radial Threads also allows the method for keeping the spiral out of the Free Area to be formalized. Because there is now a more regular guaranteed ODE interrupt than each cycle, the model can check whether the thread will pass into the Free Area before reaching the next Radial Thread, and not run the ODE if Rad-(pSize/2)/Radii < FreeRad. (pSize/2)/Radii is the maximum amount Rad can decrease between Radial Threads.

The addition of the Radial Threads also allows the control conditions to enter their final form. Theta >= 2* π is a condition that triggers when the ODE breaks as a result of violating the Theta <= 2* π domain constraint. Because this marks the completion of a cycle and an intersection with a Radial Thread, Theta and TickTheta are reset to 0 while RingRad is updated to the radius of the next ring in, denoted by RingRad – pSize/2. Theta < 2* π & TickTheta >= 2*

π/Radii is a condition that triggers when the ODE breaks as a result of TickTheta indicating an intersection with a Radial Thread. Thus it causes TickTheta to be reset so it can properly detect the next intersection. The Theta < 2* π requirement ensures that we don't call this weaker control statement when we have completed a full cycle. The remaining control condition simply cover all options not already covered, and indicates the current position is not one of note and requires no additional control.

The safety conditions for this model ensure that Theta has remained within its legal range and that the Basic Spiral is in the appropriate place relative to its angle and the current Ring. While being in precisely the correct location (as opposed to simply somewhere within the Ring) isn't necessary going forward in almost every case, it is critical that Rad is at the edge of the current Ring when Theta = 0 or Theta = 2* π to ensure a smooth, continuous transition of the spiral between one Ring and the next.

The model is now ready to begin work on the Advanced Spiral. The Basic Spiral provides a modeling framework for the general shape that the capture spiral will need to take, while the Radial Threads represent the literal physical framework that the Capture Spiral will need to attach to and are the only locations where it can adjust its direction. The full model would thus be a series of straight lines connecting the points at which the Basic Spiral crosses the Radial Threads.

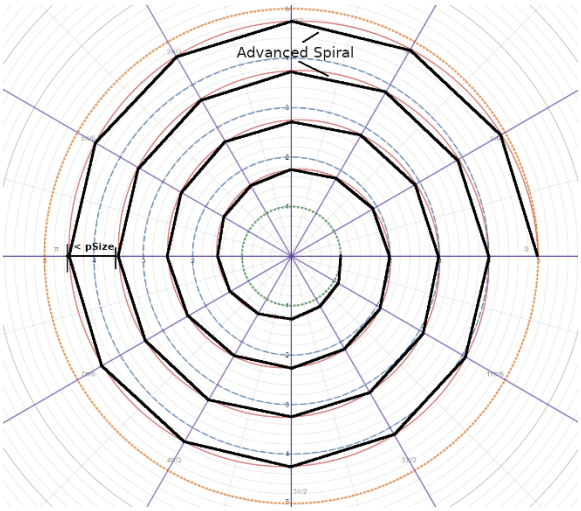


Fig 6: Advanced Spiral on Basic Spiral Framework

This has a flaw, however. Currently, we are modeling the Basic Spiral and Radial Threads with polar coordinates, where a point is defined by an angle and a distance from the origin. This allows for Radial Threads to be represented purely through single angle values, and it allows the spiral to gradually pull into the center of the web while rotating around using simple movement vectors.

Polar coordinates are not, however well suited to drawing straight lines. Lines that do not pass through the origin also do not keep a consistent or consistently changing distance from the origin, nor is there consistency between angular rate of change and the radial rate of change. Thus even modeling straight lines in polar space is extremely difficult, much less proving

anything about those lines. This, then, is where the second main step of the modeling comes into play.

2.2. Advanced Spiral Segments and Cartesian Coordinates

The process of modeling the advanced spiral has been broken down into modeling individual straight-line segments. This model is done using Cartesian coordinates and the intersection points between the Basic Spiral and the Radial Threads from the first model. These points must be converted to Cartesian coordinates between models, however.

This model uses two sets of start and end points. One set (xOutEnd, yOutEnd, xOutStart) belongs to the section of the Advanced Spiral we are currently modeling. xOutStart is the x value of the initial anchor point at which the Basic Spiral crossed a Radial Thread, while xOutEnd and yOutEnd are the x and y values, respectively, of the next anchor point where it crossed the next Radial Thread. There is no y value for the initial anchor point because this model simplifies by treating all initial points as lying long the x axis. The starting y value is known to be 0.

The other set of start and end points (xInEnd, yInEnd, xInStart) correspond to the same values for the next cycle in of the Basic Spiral, where it intersected the same two Radial Threads that defined the primary section we are modeling.

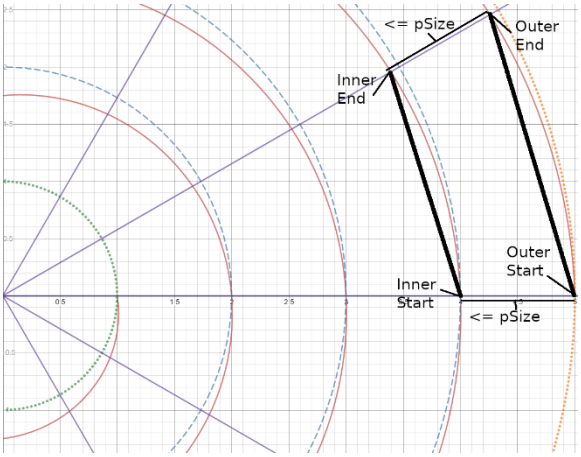


Fig 7: A Single Advanced Spiral Segment

The ODE for this model simulates concurrently tracing the straight lines from the starting points to the ending points by advancing the current outer segment location (xOut, yOut) alongside the current inner segment location (xIn, yIn). This is done at relative speeds for both outer and inner segments. We set xOut' to xOutEnd – xOutStart, yOut' to yOutEnd, xIn' to xInEnd – xOutStart and yIn' to yInEnd. Thus both segments will start being traced at the same time and finish at the same time, precisely one unit of time after the ODE began running.

This ODE is also event triggered, however the only event that will cause it to halt is the trace reaching the end positions. This is measured to be when xIn >= xInEnd or yIn >= yInEnd.

This model has no control step, as once the initial values are set there is nothing left to do but let it run to completion.

The point of this model is to ensure that the outer and inner segments of the Advanced Spiral are never more than distance pSize from each other, as well as to model the Advanced Spiral in a piecewise fashion. Any polar coordinates can be converted to the proper, simplified form prior to Cartesian translation by shifting the Theta value of all points until the Starts are at 0, and the generated model can then be shifted back into position.

Many useful properties can be assumed about the starting and ending coordinates because they are coming from the Basic Spiral. We know that adjacent Basic Spiral cycles stayed within adjacent rings with width = pSize/2, thus we know that the starting points are within distance pSize of each other, as are the end points. We know that the end points all have non-negative x and y values because with at least 4 evenly spaced Radial Threads, there cannot be more than 90 degrees between them. Thus if one lies on the x-axis the other must be somewhere in the first quadrant of the xy plane. This in turn tells us the x and y values of OutEnd are greater than those of InEnd. xOutEnd and xInEnd can be equal if Radii = 4 making the Radial Thread lie on the y-axis, however yOutEnd and yInEnd cannot be equal because pSize > 0. This is also why xOutStart and xInStart cannot be equal.

Using these properties, the model tracks the two segments concurrently to show that for every point along the outer segment of the Advanced Spiral, there is a point on the inner Advanced Spiral within distance pSize of it. The two advance at the same relative rate and the model requires that (xOut = xOutEnd -> (xIn = xInEnd & yIn = yInEnd) to ensure that the other way was also true.

3. Proof

Both the models and proofs for these models were constructed in the theorem-proving software KeyMaeraX.

3.1. Basic Spiral and Polar Coordinates

The invariants for this proof were essentially the same as the safety condition. Theta needs to remain between 0 and 2* π, inclusive, otherwise the math for calculating its position relative to RingRad begins to break down. The same is true for requiring re-confirmation that TickTheta >= 0 so there's a tighter upper bound on how long the ODE can run for.

TickTheta >= 0 is easily proven because TickTheta' = 2*π and π > 0 mean it has a strictly positive derivative and when it gets reset it gets reset to 0. The same applies for Theta >= 0. Theta <= 2* π holds because Theta starts at 0 and if Theta > 2*π it violates the loop invariant, allowing control to take over when Theta = 2* π. In that state, the only control statement with a valid condition is the one which resets Theta to 0, ensuring it is back within safe limits and only the ODE which will break if Theta > 2* π can run.

The relative position calculation itself boils down to: The distance from the current Basic Spiral location to the current Ring must be equal to how far through the current cycle the model is, scaled by the width of the Ring. This works because from the start of a cycle the Rad is being reduced at a rate balanced with the rate Theta is increasing, resulting in Rad reaching RingRad at

the same moment Theta reaches 2* π. Theta and TickTheta both halt the ODE when they reach the edge of their legal range, and control is guaranteed to reset them if they are at the upper end of their range. At the same time, control cannot reset them if they have not reached that upper end, so there is no possibility of a desync between Theta and Rad.

When the model advances to the next Ring, we know that Theta >= 2* π from the control condition and Theta <= 2* π from the invariant. Thus Theta = 2* π and Rad = RingRad = Rad – (pSize/2)*(2* π -Theta)/(2* π) = Rad - 0. Thus when RingRad is set to RingRad-pSize/2 and Theta is reset to 0 to begin the new cycle, Rad - (pSize/2)*(2* π -Theta)/(2* π) = Rad-pSize/2 = RingRad.

The FreeRad requirement is initially true because we assume Rad >= FreeRad coming in, and it remains true because the farthest Rad can possibly be pulled in before TickTheta reaches 2* π/Radii is (pSize/2)/Radii. If Rad – (pSize/2)/Radii < FreeRad then we don't run the ODE at all and Rad can't possibly advance further.

The safety conditions all follow directly from the loop invariants.

With the invariants set appropriately and none of the ODE variables advancing in a circular pattern, proving the theorem with KeyMaeraX was largely a matter of letting the math run on Quantifier Elimination.

3.2. Advanced Spiral Segments and Cartesian Coordinates

The invariants for the loop are, effectively, the model's goals. The proportion comparisons between x and y of the same lines (such as (yInEnd - yIn)/yInEnd = (yOutEnd - yOut)/yOutEnd) ensure that the trace has not deviated from the lines it is supposed to be modeling. The proportion comparison between xIn and xOut ensures that the two lines are advancing at the same proportional rate and will reach their respective ends at the same time, as the postcondition requires. All proportion invariants hold simply because the ODE progresses each variable at a constant rate proportional to the distance between where they started and where they will finish. xOut' = xOutEnd - xOutStart, yOut' = yOutEnd, xIn' = xInEnd - xInStart, yIn' = yInEnd. They all move linearly, so they all maintain their linear proportions. If the proportions stay consistent after every loop, then it stands to reason that when xOut reaches its final destination of xOutEnd, so too will xIn and yIn have reached their final destinations. Thus the safety requirement of ((xIn = xInEnd) -> (xOut = xOutEnd & yOut = yOutEnd)) holds.

The proximity invariant, (xOut-xIn)^2 + (yOut - yIn)^2 <= pSize^2, is simple in concept but was difficult for KeyMaeraX to prove. It is known that both endpoint pairs of the lines are within distance pSize of each other, and thus it follows that the distance between proportional midpoints on those straight lines cannot be greater than the distance between their endpoints. When advancing from one to the other at a constant rate, xIn must linearly transition from xInStart to xInEnd, as must all the other variables transition between their start and end points. Furthermore, they all perform those linear transitions over the same period of time. Thus, xInStart – xOutStart must then transition linearly into xInEnd – xOutEnd, and thus any midpoint xIn – xOut must be between the two ending values. If it is between them, it cannot have a greater

absolute value than either, thus the distance between two proportional midpoints of a pair of lines must not be greater than the distance between the endpoints between those lines. It is entirely possible that if the ODE were to continue on past the endpoints it would reach a state where the distance exceeded pSize, but the ODE cannot run once xIn and yIn have surpassed their endpoints, as by that point the modeling is complete. Unfortunately, calculating the potential possibilities of the interactions between four independently moving values with relatively vague constraints seems to be beyond KeyMaeraX's theorem proving abilities. The safety constraint follows directly from the loop invariant.

4. Conclusion

This problem was one that provided a conceptually daunting task with its hybridization of polar and cartesian requirements. Ultimately, tackling that task head-on proved to be fruitless and demonstrated the necessity of dividing tasks into separate pieces and addressing each in the context most suited to it.

The original goal of fully and holistically modeling the entire capture spiral as it appears in a web was not a concept that was realized, however thanks to a pivot in focus every component of the spiral was able to be modeled effectively and its safety proved. That being said, while the main deliverables of the basic and advanced spiral models were completed, a number of smaller details that were originally planned for the model had to be abandoned. The proposal for this paper stated a desire to have the maximum distance between cycles increase as the spiral approached the center of the web, since the spiral would be naturally tighter there and increased structural support from higher-density Radial Threads made such tight Capture Spiral clumping less needed. This was an idea that had to be abandoned more due to time constraints than anything else. An effective means of modeling the Advanced Spiral was not found until relatively late in the development cycle, leaving little time for additional niceties. It's an entirely reasonable feature to add to the model, however it would complicate things by no small measure, as relative strengths of capture thread and radial thread would need to be determined and the Advanced Spiral Segment model would need to be updated to account for a potentially inconsistent pSize.

5. Related Work

As was stated at the top, there is little out there on the modeling of the capture spirals themselves, however there are a number of models which address the other, more structurally robust aspects of orb-weaver webs [1]. If data were taken from one of the numerous studies about the relative strength of spider threads [2] and combined with information regarding optimal thread density for orb webs [3] then this model could be very effectively extended to adjust the spacing between cycles with proportion to the distance from the center of the web.

Even if such an optimization doesn't prove to be worth the materials saved, the data from such works could turn this model from an abstract tool into something capable of generating webs with broader parameters than simply the size of the web and a singular size for prey. If expanded with models for construction of other aspects of the orb web, this model could become a component in modeling the movements of a spider throughout the entire web creation process, something which, to my knowledge, has yet to be fully realized.

6. References

1. Y. Aoyanagi, K. Okumura
Simple Model for the Mechanics of Spider Webs
(https://journals.aps.org/prl/abstract/10.1103/PhysRevLett.104.038102)
January 20, 2010

2. F.K. Ko, J. Jovicic
Modeling of Mechanical Properties and Structural Design of Spider Web
Dept. of Materials Science and Engineering, Drexel U. March 1, 2004

3. A.L. Rypstra
Building a Better Insect Trap; An Experimental Investigation of Prey Capture in a Variety of Spider Webs (https://doi.org/10.1007/BF00349008)
Oecologia (1982) 52: 31.