A Model of a Photocopier Paper Path

Vineet Gupta
(vgupta@parc.xerox.com)
Systems and Practices Lab.
Xerox PARC
Palo Alto, Ca 94304

Vijay A. Saraswat
(saraswat@parc.xerox.com)
Systems and Practices Lab.
Xerox PARC
Palo Alto, Ca 94304

Peter Struss[1]
(struss@informatik.tu-muenchen.de)
Technical University of Munich
Orleansstr. 34
D-81667 Munich, Germany.

Abstract

We present a compositional steady-state model of paper transportation in a photocopier that is meant to support different problem solving tasks like simulation and diagnosis, and to be applicable to a variety of configurations. The model avoids making hard-wired implicit assumptions about design principles and possible scenarios, though second-order effects such as the effect of gravity, and non-zero sheet mass are ignored.

The model represents the instantaneous spatially distributed interactions between a sheet of paper and multiple transport elements (e.g., rollers) acting on it. The model can predict essential features of the motion of the sheet of paper, such as buckling and tearing. In most interesting cases, it can predict the instantaneous velocity of the leading edge of the sheet.

The framework provided is quite generic and can be used as a starting point for developing models of other transportation domains.

1 Introduction

A photocopier is a device with a clear and fixed structure that seems to lend itself to standard component-based modeling techniques. However, as already pointed out in reports about earlier modeling attempts [S^+87], this is true only if we ignore the sheets of paper that move around and interact with the copier's components, often in a way that forces us as users to sequences of recovery actions. Paper handling in a copier is an instance of a class of transportation processes whose modeling requires the modeling of the transported subject, as well.

An attempt to model the forces, motions and deformation of a sheet of paper under the influence of some rollers can lead to quite sophisticated mathematical models [SL81b, SL81a]. [GS95] presents one solution to this problem — the model deals solely with velocities, abstracting away from the forces acting on the sheet of paper. Intuitively, the information about the interaction of the various components via forces is compiled into the constraints on velocities, so reasoning with velocities is adequate for a large number of scenarios.

This paper presents a somewhat more refined model, in which some of the forces are explicitly represented. The result is a model in which the constraints are conceptually simpler — many of the equations are just free-body diagrams for the various components, while others are comparisons between magnitudes of forces. As a result, the computation of velocities is a simple exercise in Newtonian mechanics. The explicit representation of the forces also makes the model applicable in more situations, since we make fewer assumptions about the components. For example, in the velocities model, we assumed that rollers either allowed sliding of the sheet of paper or not. With the representation of friction as a force, we can now give its magnitude — a sheet does not slide if it pulled with a force less than the friction force, and it slides when pulled with a greater force.

The basic problem we address remains the same —

- modeling a device with a topology changing over time.

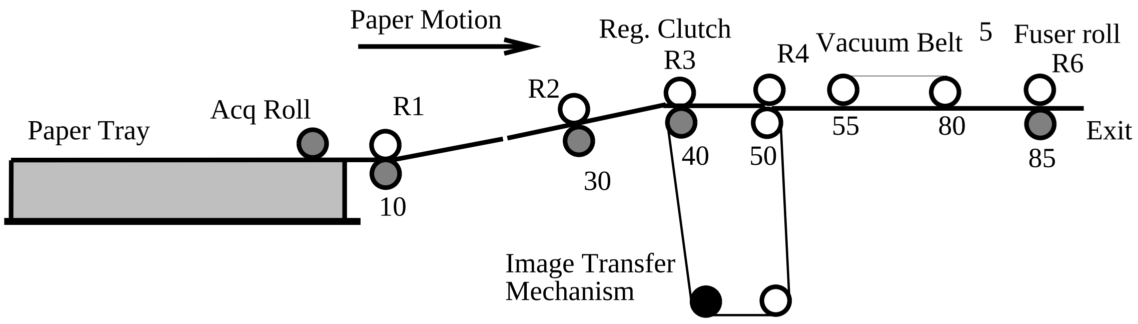

Figure 1: A simple photocopier

- finding an appropriate level of abstraction of modeling that still allows us to determine most of the interesting features, for instance buckling or tearing of paper sheets.

Furthermore, guiding principles are to

- exploit features of the physical domain for simplifying the model, but avoid hardwired implicit assumptions about possible scenarios, design principles etc.
- aim at reusable model fragments that can be further specialized and composed to cover different instances of devices, and tasks such as simulation, design and diagnosis.

The basic step we take is to consider a steady state model for the system. In steady state, the sum of the forces acting on any component is zero, as nothing is accelerating. This gives a simple set of constraints, which are solved to get the velocity of the sheet.

We model a sheet of paper as a set of interacting segments, and the sheet interacts with the transportation elements through *sheet transportation processes*. We avoid modeling the geometry of the paper path in detail — it is represented as a straight line.

We briefly introduce the domain by discussing an example of a copier paper path (Section 2) and present the intuition underlying the model in Section 3. Section 4 describes the model formally and attempts to reveal the underlying assumptions and restrictions. We apply the model to an example in Section 5, and summarize in Section 6.

2 The Domain: Paper Transportation in a Photocopier

Our problem arose in the specific domain of photocopiers, so in this section we describe the paper path of a simple photocopier, and we will illustrate the concepts developed in this paper on this machine.

In this photocopier, paper is loaded in a paper tray at the left of the machine (see Fig. 1). When a signal is received, the acquisition roll is lowered onto the paper and pulls the top sheet of paper towards the first set of rollers (1). (We will assume here that there is some mechanism to ensure that it pulls only one sheet.) After the paper is grasped by the first set of rollers, the acquisition roll is lifted, and the rollers pull the paper forward, till it reaches the registration clutch (3). This starts at a precise moment for perfect alignment, and the image is transferred onto the paper by the image transfer mechanism. The vacuum Belt transports it to the fuser roll, from where it goes to the exit.

This simple copier has most of the kinds of transportation elements that are found in the real machines — rollers, belts etc. Several interesting situations in a real copier are illustrated here. The sheet can be under more than one driven roller (rollers number 1 and 2), the rollers can change speeds (3), and there is a belt (5), which has low sliding friction. In addition, rollers have different speeds, for example the fuser moves the paper faster than the belt.

We assume that the paper and the transportation elements have negligible inertia. Hence when a sheet comes into contact with a roller, it is instantaneously accelerated, and when it is in contact with the belt, the small friction force is sufficiently large to move the paper at the same speed as the belt (unless some other transportation element prevents this). This assumption of negligible inertia means that the time required for acceleration is very small, and from our knowledge of the design of the paperpaths, this seems to be the case. However this assumption needs further validation.

Our model is sufficiently general to cover several scenarios that may not arise in a designed copier. For example, our model is able to predict that if we had two successive rollers moving in opposite directions, then the paper would tear or slip. If the back roller moves faster than the front roller, the paper can buckle. This generality is necessary from several points of view. Firstly the designer may have made an error, in which case simulating with this model would reveal the error. Secondly something may have gone wrong in the machine, then this prediction would be required for diagnosis. And in any case this is necessary from general design principles, since we want our model to work with as few assumptions as possible, and assuming good design is certainly a strong assumption.

3 Modeling the Paper Transportation — The Intuition

Before we present the formal model of the paper transportation, we try to convey its intuitive background.

In principle, a sheet transported in a copier corresponds to a 2-dimensional surface being deformed and moving around in a 3-dimensional space in a continuous way (unless it tears). For the purposes of our model, we make a big simplification — we model the sheet by its cross-section, moving along a straight line. This means we assume that the sheet does not deviate from its intended path. Also we do not model many two-dimensional phenomena like skewing of the sheet and transverse buckling (i.e. shortening perpendicular to the direction of motion). We do model standard buckling (shortening along the direction of motion) by measuring the decrease in length. Branching of the paper path can be handled as in [GS95], and we will omit it to maintain simplicity.

The overall motion of the transported sheet, as well as potential deformations of the sheet, such as buckling or tearing, result from its inertia and interaction of the sheet with one or more transportation elements (such as belts, pairs of roller, etc), and other potential means (such as gravity). Each interaction of this kind is *local*, and occurs due to objects applying some forces on each other. The magnitude of the forces is influenced by the type and actual properties of two basic classes of objects: the *sheet*, which can be a transparency, paper, etc. and accordingly have different properties (smoothness, stiffness, strength etc.), and the *transportation elements* of different kinds: for instance, a roller can apply a large friction force on the sheet, whereas a belt does not — the paper starts slipping instead. All the forces together move the sheet of paper, and determine its speed.

Our basic idea is to directly represent these localized interactions between the transportation elements and the sheet, and to infer the motion of the entire sheet from these local interactions. A sheet is modeled as consisting of a sequence of contiguous segments: a *contact* segment (the portion of the sheet in contact with a transportation element) or an internal segment (the portion of a sheet between contact segments), or left- or right-tail segments. The number and extent of these segments varies from instant to instant. The free-body diagrams for each of these segments are analyzed. Since we are interested in a steady state (equilibrium) analysis, the resulting force on each segment is equated to zero.

A second important source of constraints are limits on the forces acting on the sheet of paper. A roller cannot pull the paper with more force than the friction force between it and the paper. A paper can transmit forces below a certain magnitude only — it buckles or tears beyond these limits. This helps in the determination of force magnitudes, since in most situations at least some forces will be at these limit values.

We now consider the model in detail.

4 Description of the Model

4.1 Modeling language.

We use a continuous temporal logic Hybrid Concurrent Constraint Language [GJSB95b, GJSB95a] (Hybrid tcc) as the formal notation for our model. The model is given here as a set of model fragments or *agents*, represented as $g :: A$, where g is an atomic formula, and A is an agent. Agents are viewed as imposing constraints on the evolution of the system over the real line, called a *run*. Primitive constraints are simple instantaneous assertions of the form `X + Y = 5`, or `(d/dt)(X) = Y+5`. The agent c (for c a primitive constraint) imposes just the constraint c at the current time instant. The agent **if** c **then** A imposes the constraints of A provided that the rest of the system imposes the constraints c at the current time instant. The agent **if** c **else** A imposes the constraints of A unless the rest of the system imposes the constraints c at the current time instant.[2] **new** X **in** A imposes the constraints of A, but hides the variable X from the other agents. The agent A, B imposes the constraints of both A and B. The agent **hence**A imposes the constraints of A at every time instant after the current one. If $g :: A$ is a model fragment, then g is a primitive constraint, and adding this constraint corresponds to adding the constraints of A.

This denotational way of looking at model fragments is complemented by an operational view. Often both views are helpful in understanding a model fragment. The basic idea in the operational view is that of a network of agents interacting with a shared store of primitive constraints at each time instant on the real line. The agent c is viewed as adding c to the store at the current time instant. The agent **if** c **then** A behaves like A if the current store entails c. The agent **if** c **else** A behaves like A if the current store on quiescence does *not* entail c. **new** X **in** A starts A but creates a new local variable X, so no information can be communicated on it outside. The agent A, B behaves like the simultaneous execution of both A and B. The agent **hence**A behaves like A executing at every real time instant after the current one. If $g :: A$, then whenever g is present in the store, A immediately starts execution. If g has a list of parameters, this is handled by textual substitution.

Several idioms are definable directly in Hybrid tcc. For instance, **default** c, an assertion that requires c to hold unless it can be shown not to to hold, is merely **if** $not(c)$ **else** c. `always A`, an assertion that requires A to hold at every time instant including the current one, is merely A, **hence**A. In the following, we will use two other constructs definable in Hybrid tcc: **do** A **watching** c and **whenever** c **do** A. Intuitively, the first executes A, aborting it at the first time instant at which c is true[3]. The second starts the execution of A at the first time instant at which c holds.

[2]This construct is motivated by defaults from the default logic of Reiter [Rei80], and corresponds to the formula $: \mathbf{M}\neg c / A$. See [SJG95] for details.

[3]Note that since A may contain **hence**B sub-terms, its execution may extend across time.

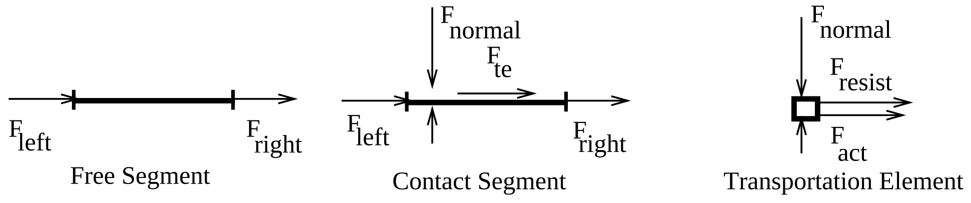

Figure 2: Forces on each element.

The constraint system allows the assertions of primitive atoms such as `transportation(X)`, arithmetic (in)equalities and propositional logic formulas built with them, and feature structures (attribute-value lists). A feature structure is a finite mapping associating each of a given set of attributes with a given value. Given a feature structure `L` and attribute `a`, the term `L.a` refers to the value associated with `a` in `L` — if there is such a value. Similarly, the term `L[position = P]` represents the feature structure `T` which identical to `L`, except that it takes on the value `P` at attribute `position`. We write `L[position = (20,20)][surfacetype = rubber]` as `L[position = (20,20), surfacetype = rubber]`.

We follow Prolog convention for the naming of constants and variables (the former begin with a lower case letter; the latter with an upper case.

4.2 Transportation Elements

We now turn to the models of the objects and processes involved in paper transportation.

The paper path is represented as an initial section of the real line. We will consider intervals on this path — each interval is represented as a pair of real numbers denoting the upper and lower bounds of the interval. The lower bound of an interval is denoted $lb(I)$, and the upper bound as $ub(I)$. The intersection $I \cap J$ is defined for intervals in the usual way $I \cap J \overset{d}{=} (max(lb(I), lb(J)), min(ub(I), ub(J))$. We define $I < J$ by $ub(I) < lb(J)$, so I and J are disjoint, and I precedes J. The length of an interval is $|I| = ub(I) - lb(I)$ — we will always consider intervals with length > 0.

We will first describe the components of the copier that drive the sheets along the paper path. Each transportation element has a location `Loc` which is a non-empty interval on the paperpath. For the purposes of this paper, we need only model transportation elements that are fixed in space. This is asserted by the formula **always** `T.Loc = Init.Loc`. A component may not always be active — the variable `T.engaged` specifies whether it is currently acting or not. We assume that transportation elements do not overlap on the paper path.

Each transportation element has a property `surfacetype` which determines the coefficient of friction. It also has a normal force, $\texttt{F}_{\texttt{normal}}$, which may be applied by a spring, or vacuum or other means. It may depend on the thickness of the paper, but here we assume it constant. When multiplied by the coefficient of friction, this gives the friction force.

The variable $\texttt{V}_{\texttt{nom}}$ denotes the *velocity* with which the component would carry the paper along the interval `Loc` in the absence of all other forces. It is assumed to be constant along `Loc`, and is determined by the speed of the motor that drives it. Although for most components $\texttt{V}_{\texttt{nom}}$ is fixed, there are exceptions: the registration clutch has a variable velocity. The transportation element can be forced to move faster or slower than its nominal velocity by application of a sufficient force — $\texttt{F}_{\texttt{fast}}$ is the force required to move an element faster than the nominal velocity, while $\texttt{F}_{\texttt{slow}}$ gives the force required to slow it down. We assume these are constants, as they measure the resistance offered by the element in each direction. Finally, the force exerted on the transportation element is denoted by $\texttt{F}_{\texttt{act}}$. The freebody equation relates this force to the force exerted by the motor on the element, $\texttt{F}_{\texttt{resist}}$. Figure 2 shows the freebody diagrams for transportation elements and segments.

```
transportationElement(T, Init) ::
    always (transport(T),
            T.Loc = Init.Loc, T.surfacetype = Init.surfacetype,
            $T.F_{fast} = Init.F_{fast}$, $T.F_{slow} = Init.F_{slow}$,
            $T.F_{normal} = Init.F_{normal}$,
            $T.F_{act} + T.F_{resist} = 0$,
            $-T.F_{fast} \leq T.F_{resist} \leq T.F_{slow}$).
```

Various kinds of components can be represented as refinements of this generic transportation element, that satisfy certain additional conditions (see section 4.5). Belts usually have low friction coefficients and normal forces, with high $F_{fast}$and F_{slow}. For a roller with a one-way clutch, F_{fast}would be quite small, as the roller is free to spin in that direction, whereas $F_{slow}$is very high, since the roller would have to break, or the motor would burn in order to move it slower. Therefore the agent **if** $R.F_{act}= R.F_{slow}$ **then** R.broken can be used to state that under these conditions the roller is broken. Note that we did not mention the variety of different physical mechanisms that are used in such components, for instance normal spring or vacuum forces. If these details matter, e.g. for diagnostics, the components classes could be further specialized.

4.3 Sheet and Sheet Segments

The second basic kinds of object are the *sheets* that are being moved around. A sheet has properties length, width, thickness and surfacetype. Each sheet also has a minimum force required to tear it, F_{tear}, and the modulus of elasticity which determines when it would buckle. Our current model treats these properties as homogeneous across the whole sheet — this restriction may be dropped in the future in order to cope with media such as transparencies with a white strip. Note that everything is inside a **do** ... **watching** S.torn, so when a sheet is torn, this agent is aborted, as we do not wish to continue modeling a torn sheet.

A sheet has a location S.Loc, which is an interval on the paper path. The sheet may participate in an arbitrary number of sheetTransportation interactions arising from the transportation elements it is currently in contact with. The model fragment below determines the number and nature of *segments* that the sheet may be thought of as being divided into, at any instant. The interval on a sheet that is part of a sheet transportation process is a contactSegment. An internalSegment is an interval between two adjacent contactSegments — that is, it is the interval between two contactSegment intervals I, J such that there is no contactSegment K between them. The part of the sheet, if any, beyond the rightmost contactSegment, is a rightTailSegment, similarly we have a leftTailSegment. Note that the segments are "virtual" parts — their number and extent is determined "instantaneously" based on the current position of the sheet and the transport elements, and varies from instant to instant.

Since transportation elements are assumed to have disjoint positions on the path, the above segments are disjoint, and partition the sheet (Fig. 3). The model treats these segments as the primitive constituents of a sheet. The motion and condition of a sheet is determined by the motion and condition of its segments. For instance, as soon as one of the segments of the sheet is determined to be torn, the sheet is determined to be torn. In the following, we use another Hybrid tcc construct: **forall** I:c. A. This behaves as the conjunction of A[t/I] for all terms t such that the store entails c[t/I]. To ensure boundedness of the computation, it is necessary to establish that the number of such terms is bounded; this can be done for the model below.

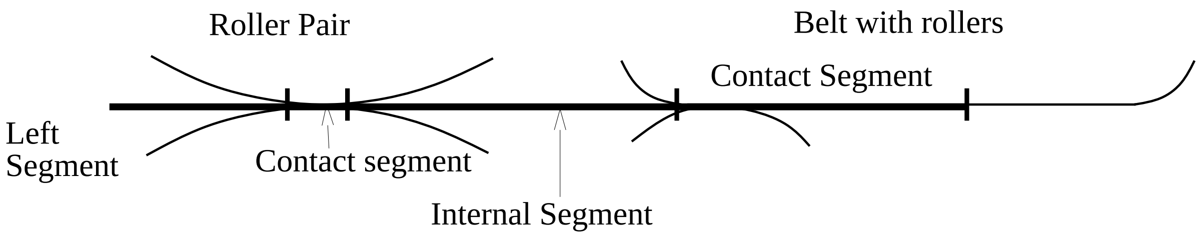

Figure 3: Segments of a sheet.

```
sheet(S, Init) ::
    S.Loc = Init.Loc,
    do always
        (sheet(S),
         S.width = Init.width, S.length = Init.length, S.thickness = Init.thickness,
         S.elasticity = Init.elasticity, S.surfacetype = Init.surfacetype,
         S.strength = Init.strength,
         if ∃I.(S.I.condition = tearing) then S.torn,
         forall T : transport(T). if (T.engaged, | T.Loc ∩ S.Loc | > 0) then contact(S, S.Loc ∩ T.Loc),
         forall I : contact(S, I). contactSegment(S, I),
         forall I : contact(S, I). forall J : contact(S, J).
           if I < J then if ∃K.(contact(S, K), I < K < J) else internalSegment(S, (ub(I), lb(J))),
         forall I : contact(S, I). if ∃K.(contact(S, K), K < I) else leftTailSegment(S, (lb(S.Loc), lb(I))),
         forall I : contact(S, I). if ∃K.(contact(S, K), I < K) else rightTailSegment(S, (ub(I), ub(S.Loc))),
         (d/dt)(lb(S.Loc)) = S.vel.lb(S.Loc),
         (d/dt)(ub(S.Loc)) = S.vel.ub(S.Loc))
     watching S.torn.
```

Segments. We now consider the model fragments for segments.

Each segment is a `sheetSegment`. Each segment has some force required to buckle it, calculated according to Euler's formula[4]. It has a condition, which may be straight, tearing or buckled. By default, we state that segments are straight. In the following each point `x` of the sheet has a velocity `S.vel.x`, and constraints are imposed on these. Each point of the sheet also has forces acting on it from the left ($\texttt{S.x.F}_{\texttt{left}}$) and the right ($\texttt{S.x.F}_{\texttt{right}}$). The segment constraints relate the forces with the velocities.

```
sheetSegment(S, I) ::
    Segment(S, I),
    S.I.F_buckle = (π^2/3) × S.width × (S.thickness)^3 × S.elasticity/ | I | ^2,
    default S.I.condition = straight.
```

As indicated above, we distinguish between different classes of segments depending on how they are related to the transportation elements. `contactSegments` are the segments directly under the influence of a transportation element. This influence will be modeled in the `sheetTransportation` process described in Section 4.4. The only thing we model here is that the two ends of a contact segment have the same velocities and that no buckling or tearing occurs — actually this might be violated on a belt, at least under fault conditions. Also, if the `contactSegment` is at the end of a sheet, we assert that there are no forces acting on the edge of the sheet.

[4]Our analysis of buckling is very simplified — we assume that the sheet buckles uniformly along its width, and can still apply the force $\texttt{F}_{\texttt{buckle}}$ at its ends.

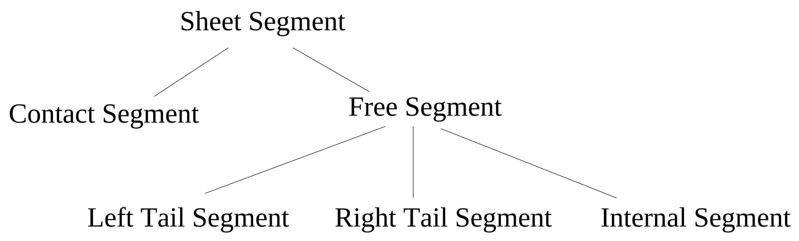

Figure 4: Class hierarchy of sheet segments.

```
contactSegment(S, I) ::
    sheetSegment(S, I),
    $S.vel.(lb(I)) = S.vel.(ub(I))$, $S.I.condition = straight$,
    if $lb(I) = lb(S.Loc)$ then $S.(lb(I)).F_{left} = 0$,
    if $ub(I) = ub(S.Loc)$ then $S.(ub(I)).F_{right} = 0$.
```

The other segments are `freeSegments` — segments on which no transportation element acts. The first 3 constraints correspond to the freebody equation for the segment and its endpoints. The forces can be between certain magnitudes only, as the sheet would otherwise buckle or tear. The amount of buckling, if any, is given by `S.I.buckle_amt` — the rate of buckling is given by the difference in velocities of the two endpoints. We assume that a buckled segment transmits a force equal to $S.I.F_{buckle}$, this assumption will be abandoned in future models.

If the segment is straight, then the forces transmitted by it are related to the relative velocities of the endpoints, in the obvious way. If it is not buckled, and the right velocity is determined to be greater than the left, then it tears.

```
freeSegment(S, I) ::
    (sheetSegment(S, I),
     $S.(lb(I)).F_{left} + S.(ub(I)).F_{right} = 0$,
     $S.(ub(I)).F_{left} + S.(ub(I)).F_{right} = 0$,
     $S.(lb(I)).F_{left} + S.(lb(I)).F_{right} = 0$,
     $-S.I.F_{buckle} \leq S.(ub(I)).F_{right} \leq S.strength$,
     if $S.I.buckle\_amt > 0$ then $(S.I.condition = buckled,\ S.(ub(I)).F_{right} = -S.I.F_{buckle})$,
     default $S.I.buckle\_amt = 0$,
     if $(S.I.buckle\_amt > 0 \vee S.vel.(lb(I)) > S.vel.(ub(I)))$ then
        $(d/dt)(S.I.buckle\_amt) = S.vel.(lb(I)) - S.vel.(ub(I))$,
     if $S.I.condition = straight$
       then $(S.(ub(I)).F_{right} = S.strength \Rightarrow S.vel.(lb(I)) \leq S.vel.(ub(I))$,
             $S.(ub(I)).F_{right} = -S.I.F_{buckle} \Rightarrow S.vel.(lb(I)) \geq S.vel.(ub(I))$,
             $-S.I.F_{buckle} < S.(ub(I)).F_{right} < S.strength \Rightarrow S.vel.(lb(I)) = S.vel.(ub(I)))$,
     if $(S.I.buckle\_amt = 0,\ S.vel.(lb(I)) < S.vel.(ub(I)))$ then $S.I.condition = torn)$.
```

The first kind of free segments are `tailSegments`, which have a transportation element acting on only one side. Since nothing acts on the left edge, the force on the right hand side must be either 0 or F_{buckle}, depending on whether the segment is straight or buckled. `rightTailSegment` is analogous. In the current model we do not consider situations when no transportation element is acting on the sheet (e.g. at the output). This would require considering gravity and inertia, both of which are avoided.

```
leftTailSegment(S, I) ::
    freeSegment(S, I),
    if S.I.condition = buckled then
      if ∃T.[transport(T) ∧ T.engaged ∧ ub(T.Loc) = lb(I) ∧ $T.V_{nom} \geq 0$] then S.vel.lb(I) = 0,
    if S.I.condition = straight then $S.(lb(I)).F_{right} = 0$.
rightTailSegment(S, I) ::
    freeSegment(S, I),
    if S.I.condition = buckled then
      if ∃T.[transport(T) ∧ T.engaged ∧ lb(T.Loc) = ub(I) ∧ $T.V_{nom} \leq 0$] then S.vel.ub(I) = 0,
    if S.I.condition = straight then $S.(ub(I)).F_{left} = 0$.
```

Finally, there are the free segments with sheet transportations on both sides, the `internalSegments`. These impose no additional conditions.

```
internalSegment(S, I) :: freeSegment(S, I).
```

4.4 Sheet Transportation

This process determines the interaction between a transportation element and a sheet. It is set up by `interact`, which says that if a transportation element shares an interval with a sheet and is engaged, then there is an interaction.

```
interact(S) ::
    always forall T : transport(T).
      if (T.engaged, | T.Loc ∩ S.Loc | > 0) then sheetTransportation(S, T, T.Loc ∩ S.Loc).

sheetTransportation(S, T, I) ::
    new $F_{friction}$ in
      (I = S.Loc ∩ T.Loc,
       $F_{friction}$ = mu(S.surfacetype, T.surfacetype) × $T.F_{normal}$,
       $S.I.F_{te} + T.F_{act} = 0$,
       $S.(lb(I)).F_{left} + S.(ub(I)).F_{right} + S.I.F_{te} = 0$,
       $|S.I.F_{te}| \leq F_{friction}$,
       S.vel.(ub(I)) > T.vnom ⇒ $S.I.F_{te} = \max(-T.F_{fast}, -F_{friction})$,
       S.vel.(ub(I)) < T.vnom ⇒ $S.I.F_{te} = \min(T.F_{slow}, F_{friction})$,
       $\max(-T.F_{fast}, -F_{friction}) < S.I.F_{te} < \min(T.F_{slow}, F_{friction})$ ⇒ S.vel.(ub(I)) = T.vnom).
```

The friction coefficient between the sheet and the transportation element is determined by their surfacetypes, here the function would simply be a table lookup. The friction force is the product of the normal force and the friction coefficient. F_{te} is the force exerted by the transportation element on the sheet, this is equal and opposite to the force exerted by the sheet on the element. The next equation is a freebody equation for the sheet. Finally, we note that the transportation element can interact with the sheet only through friction, hence the force exerted by it on the sheet cannot be greater than the friction force. Also, if the velocity differs from the nominal velocity of the transportation element, then the force applied by it must be at one of the two extremes possible, as either the sheet is slipping, or the roller is broken or is providing the maximum resistance possible.

4.5 The Model for the Example Copier

We now give a model for the copier described in section 2. All lengths and distances are given in centimetres, forces in newtons and elasticity in gigapascals. Note that values are only representative — they are *not* derived from real data. Each physical component is represented as a binary predicate; its first argument is the name of the component being defined, and the second an attribute-value list containing the specification of values. `cvElement` is a transportation element with a constant velocity. A `Roller` is a kind of `cvElement` which cannot rotate faster or slower than its nominal velocity without breaking. A `clutchedRoller` can rotate faster than its nominal velocity due to a one-way clutch. A belt has very low friction. A `fuserRoll` has high normal force, as it fuses the toner into the paper.

```
cvElement(R, Init) :: transportationElement(R, Init), always (R.vnom = Init.vnom, R.engaged).
roller(R, Init) ::
    cvElement(R, Init[surfaceType = rubber, $F_{fast}$ = 1000, $F_{slow}$ = 1000, $F_{normal}$ = 300]),
    if (R.$F_{act}$ = R.$F_{slow}$ ∨ R.$F_{act}$ = −$F_{fast}$) then R.broken.
clutchedRoller(R, Init) ::
    cvElement(R, Init[surfaceType = rubber, $F_{fast}$ = 3, $F_{slow}$ = 1000, $F_{normal}$ = 200]),
    if R.$F_{act}$ = R.$F_{slow}$ then R.broken.
belt(R, Init) ::
    cvElement(R, Init[surfaceType = steel, $F_{fast}$ = 10000, $F_{slow}$ = 10000, $F_{normal}$ = 120]).
fuserRoll(R, Init) ::
    cvElement(R, Init[surfaceType = rubber, $F_{fast}$ = 1000, $F_{slow}$ = 1000, $F_{normal}$ = 1000]).
regClutch(R, Init) ::
    transportationElement(R, Init), always R.engaged,
    whenever R.on do do always R.vnom = Init.vnom watching R.off,
    whenever R.off do do always R.vnom = 0 watching R.on.
```

An example of a thick A4 size glossy sheet is given below.

```
a4GlossySheet(S) ::
    sheet(S, [length = 21, width = 29.7, thickness = 0.015, elasticity = 15,
              surfaceType = glossy, strength = 500, Loc = (0, 21)]).
```

The whole model is presented below in terms of the component descriptions and the domain descriptions given in previous subsections. The copier's paperpath has 6 rollers, each having the properties shown. Every time the copier receives a signal `Sync`, the acquisition mechanism (which is not modeled here) places a new sheet `S` at a location $(0, 21)$, and then it is transported to the end of the path. The construct **wait** `t` **do** `A` waits `t` seconds and then starts `A`.

```
copierModel(Sync) :: new (R1, R2, R3, R4, R5, R6) in
    (roller(R1, [$V_{nom}$ = 30, Loc = (10, 10.5)]),
     roller(R2, [$V_{nom}$ = 30, Loc = (30, 30.2)]),
     regClutch(R3, [$V_{nom}$ = 30, Loc = (39.9, 40.1),
                    surfaceType = rubber, $F_{fast}$ = 1000, $F_{slow}$ = 1000, $F_{normal}$ = 300]),
     clutchedRoller(R4, [$V_{nom}$ = 0, Loc = (50, 50.1)]),
     belt(R5, [$V_{nom}$ = 30, Loc = (55, 80)]),
     fuserRoll(R6, [$V_{nom}$ = 35, Loc = (85, 86)]),
```

always (f(rubber, glossy) = 0.8, f(steel, glossy) = 0.1),
always if Sync **then** (**new** S **in** (a4GlossySheet(S), interact(S)),
R3.off, **wait** .8 **do** R3.on)).

5 Using the model

The obvious use of our model is to execute it — it then simulates the photocopier. However, we can exploit the fact that it has been presented in a formal logical language to use it for various other purposes.

Formal verification. We would like to know whether our model satisfies certain intuitive conditions. For example, we would like to state that the amount of buckling for any sheet of paper will be at most 5 cm. This can be expressed as:

always forall S : sheet(S). **forall** I : segment(S, I). S.I.buckle_amt $\leq$ 5

Other useful properties are that a sheet will never tear, or that two successive sheets will never overlap. These can be expressed similarly:

always forall S : sheet(S). not(S.torn)..
always forall S1 : sheet(S1). **forall** S2 : sheet(S2). | S1.Loc $\cap$ S2.Loc | = 0..

Let the domain theory given in the previous section be denoted θ, and the model in section 4.5 be M. We would now like to ensure that any run of M, θ satisfies a given property ϕ. Since ϕ is also a program, the runs satisfying ϕ are exactly the runs of ϕ. Thus we can say that M, θ satisfies ϕ if any run of M, θ is also a run of ϕ — this is formally denoted $M, \theta \vdash \phi$. This can be established by model checking or theorem proving techniques [GJSB95b, GJSB95a].

Other uses. Note that the third property is not true for our model — it is violated if successive `Sync`'s come close to each other. The model can now be used to prove a condition on the minimum time interval between successive `Sync`'s. A generalization of this technique can be used for *control code generation*. We can run the model backward for scheduling by abduction, and can use it for diagnosis by building fault conditions into the model fragments.

As an example to illustrate the use of our model, we apply it to the scenario when the sheet has reached the registration clutch, so is at the location $(19, 40)$, after traveling for $19/30$ seconds. Since the Registration Clutch will go on only after 0.8 seconds, its nominal velocity is zero. We can show that this would be the situation by proving

$$M, \theta \vdash \textbf{whenever } \texttt{Sync} \textbf{ do wait } 19/30 \textbf{ do } (\textbf{new } S \textbf{ in } (S.\texttt{Loc} = (19, 40)), \texttt{R3.off})$$

Now from the right-hand side we can prove that `S.vleft = 30, S.vright = 0, (d/dt)(S.(30.2, 39.9).buckle_amt) = 30`. We show the deduction informally.

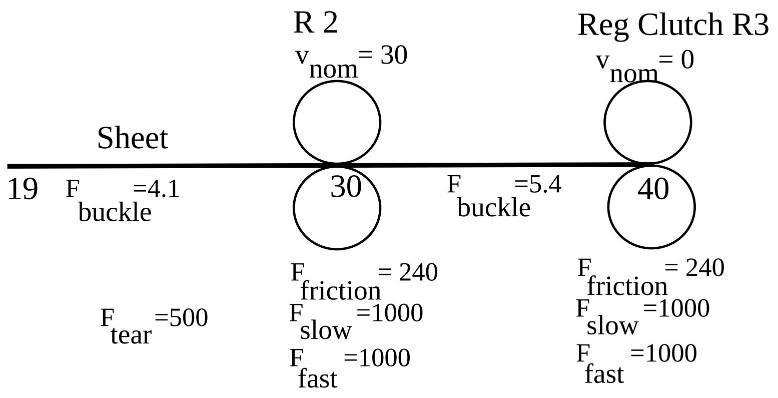

Figure 5: An example scenario.

The process `interact` sets up two sheet transportation processes at $(30, 30.2)$ and $(39.9, 40)$, as these are the only two elements overlapping the sheet. The process sheet then sets up contactsegments at $(30, 30.2)$ and $(39.9, 40)$, which leads to a left tail segment at $(19, 30)$ an internal segment at $(30.2, 39.9)$.

The sheetSegments compute the buckling forces, these are 4.1 and 5.4 respectively. Thus $-4.1 \leq S.30.\mathtt{F}_{\mathtt{right}} \leq 500$, and $-5.4 \leq S.(39.9).\mathtt{F}_{\mathtt{right}} \leq 500$. Further the second contact segment asserts that $S.40.\mathtt{F}_{\mathtt{right}} = 0$, while the left tail segment asserts that $S.19.\mathtt{F}_{\mathtt{left}} = 0$. The freebody constraint on the left tail segment now gives $S.30.\mathtt{F}_{\mathtt{right}} = 0$, which further entails $S.30.\mathtt{F}_{\mathtt{left}} = 0$.

The sheetTransportation processes set up further constraints. As `mu` = 0.8, and $\mathtt{F}_{\mathtt{normal}} = 300$, the friction force is 240 for both. From the freebody equations for each of the contact segments, using the constraints given above, we have $S.(30.2).\mathtt{F}_{\mathtt{right}} + S.(30, 30.2).\mathtt{F}_{\mathtt{te}} = 0$ and $S.(39.9).\mathtt{F}_{\mathtt{left}} + S.(39.9, 40).\mathtt{F}_{\mathtt{te}} = 0$. From the freebody equations for the points 30.2 and 39.9, we immediately get $S.(30.2).\mathtt{F}_{\mathtt{left}} = S.(30, 30.2).\mathtt{F}_{\mathtt{te}}$ and $S.(39.9).\mathtt{F}_{\mathtt{right}} = S.(39.9, 40).\mathtt{F}_{\mathtt{te}}$.

Now from the constraints imposed by sheetTransportation process on $\mathtt{F}_{\mathtt{te}}$, we get

$$-240 \leq S.(30, 30.2).\mathtt{F}_{\mathtt{te}}, S.(39.9, 40).\mathtt{F}_{\mathtt{te}} \leq 240$$

. Combining this with the constraints on $S.(39.9).\mathtt{F}_{\mathtt{right}}$, we get $-5.4 \leq S.(39.9, 40).\mathtt{F}_{\mathtt{te}} \leq 240$. This means that $S.vel.(39.9) \leq 0$, since otherwise $S.(39.9, 40).\mathtt{F}_{\mathtt{te}} = max(-240, -1000)$, which is impossible. A similar analysis yields $-240 \leq S.(30, 30.2).\mathtt{F}_{\mathtt{te}} \leq 5.4$.

If $S.(39.9, 40).\mathtt{F}_{\mathtt{te}} = 240$, then by the implications in the freeSegment, we have $S.vel.(30.2) = S.vel.(39.9) \leq 0$. This means $S.(30, 30.2).\mathtt{F}_{\mathtt{te}} = 240$, but then the freebody equation for the internal segment would not be satisfied. Thus we have $S.(39.9, 40).\mathtt{F}_{\mathtt{te}} \neq 240$, so $S.vel.(39.9) = 0$.

Now from the constraints on $S.(30, 30.2).\mathtt{F}_{\mathtt{te}}$, we have $S.vel.(30.2) \geq 30$. If it is greater than 30, then the implications in the freeSegment would entail $S.(30.2).\mathtt{F}_{\mathtt{left}} = 5.4$, which means $S.(30, 30.2).\mathtt{F}_{\mathtt{te}} = 5.4$, which is contradicted by the implications in the sheetTransportation process. Thus $S.\mathtt{v}el.(30.2) = 30$.

Now we know that the paper is buckling the two rollers at a rate of $30 - 0 = 30$. The model may place additional bounds on the amount of buckling possible, then we could find out which of the events would occur first — a jam due to excessive buckling, the switching on of the registration clutch or the passing of paper through the roller at position (30, 30.2). If a jam occurs, we know that the value 0.8 for the wait for `R3.on` is too much, and has to be reduced.

6 Summary and Discussion

The work presented here is a first step towards a declarative and general model of paper handling in a copier. The concepts, however, are more general and should be useful in other transportation processes (e.g. in textile manufacturing or printing). The key point is splitting the overall process into locally acting transportation processes, which are then related by forces transmitted by the transported object. Particular properties, such as elasticity, of these objects can then be expressed at these points.

The model provides a good coverage of the actual and conceivable scenarios in paper handling, with some limitations pointed out in section 4. This has been achieved at a fairly abstract qualitative level of representation that refers to forces at equilibrium only.

By assuming a steady state situation, we still avoid a detailed study of acceleration, and consequent phenomena like impulses, which could lead to tearing etc. However, a "gold-standard" model that includes these has to be related to our model in order to reveal its assumptions and limitations (e.g. using the theory of model transformation and simplification of [Str92]).

Acknowledgements. We gratefully acknowledge the discussions with David Bell, Danny Bobrow, Brian Falkenhainer, Markus Fromherz and Mark Shirley, and thank them for their valuable comments. The first two authors would like to acknolwedge partial support from NASA, and the third author would like to thank the International Computer Science Institute at Berkeley and Xerox PARC for support of his research visit in Berkeley.

References

[GJSB95a] V. Gupta, R. Jagadeesan, V. Saraswat, and D. Bobrow. Programming in Hybrid Constraint Languages. Technical report, Xerox PARC Systems and Practices Lab, February 1995. forthcoming.

[GJSB95b] V. Gupta, R. Jagadeesan, V. Saraswat, and D. Bobrow. Reactive Computing with Continuous Change. Technical report, Xerox PARC Systems and Practices Lab, March 1995. forthcoming.

[GS95] Vineet Gupta and Peter Struss. Modeling a copier paper path : A case study in modeling transportation processes. In *Proceedings of the Qualitative Reasoning Workshop 1995*, May 1995.

[Rei80] Ray Reiter. A Logic for Default Reasoning. *Artificial Intelligence*, 13:81 – 132, 1980.

[S$^+$87] Jeff Shrager et al. Issues in the Pragmatics of Qualitative Modeling: Lessons Learned from a Xerographics Project. *Comm. of the ACM*, 30:1036–1047, 1987.

[SJG95] V. A. Saraswat, R. Jagadeesan, and V. Gupta. Default Timed Concurrent Constraint Programming. In *Proceedings of Twenty Second ACM Symposium on Principles of Programming Languages, San Francisco*, January 1995.

[SL81a] T.C. Soong and C. Li. The Rolling Contact of Two Elastic-layer-covered Cylinders Driving a Loaded Sheet in the Nip. *J. of Applied Mechanics*, 48:889–894, 1981.

[SL81b] T.C. Soong and C. Li. The Steady Rolling Contact of Two Elastic Layer Bonded Cylinders with a Sheet in the Nip. *Int J. of Mech. Sci.*, 23:263–273, 1981.

[Str92] Peter Struss. What's in SD? Towards a Theory of Modeling for Diagnosis. In W. Hamscher, L. Console, and J. de Kleer, editors, *Readings in Model-based Diagnosis*. Morgan-Kaufmann, 1992.