Mapping Communicative Goals into Conceptual Tasks to Generate Graphics in Discourse*

Stephan Kerpedjiev♣
MAYA Viz♣
2100 Wharton Street
Pittsburgh, PA 15203
kerpedjiev@mayaviz.com

Steven F. Roth♣♦
Robotics Institute♦
Carnegie Mellon University
Pittsburgh, PA 15213
roth@mayaviz.com

ABSTRACT
We address the problem of realizing communicative plans in graphics. Our approach calls for mapping communicative goals to conceptual tasks and then using task-based graphic design for selecting graphical techniques. In this paper, we present the mapping rules in several dimensions: data aggregation and selection, task synthesis, and task aggregation. Those rules have been incorporated in AutoBrief, a research system for multimedia explanation.

Keywords
Data graphic design, communicative plans, conceptual tasks

1. INTRODUCTION
Visualizations are used in narratives, arguments, explanations and other communicative genres to succinctly convey complex relations or to organize a large amount of information. Such presentations are planned to achieve communicative goals (e.g., the user believes that *insufficient airport capacity is the cause for some late cargo*). Communicative planning involves reasoning about communicative goals, user's beliefs, and information about the domain, in order to select a sequence of communicative acts (such as asserting a proposition) that achieves the goals [9]. The realization of communicative plans has traditionally been studied in the context of natural language generation. In this paper, we address the realization of communicative goals in graphics.

The problem is that automated graphic design has been studied exclusively from the point of view of data exploration, where the problems are specified in terms of data sets and the tasks that the user needs to perform [3, 12]. The graphic designer selects graphical techniques (e.g., axes, bars, lines, and color) that make the execution of the tasks efficient. In general, tasks are performed efficiently when the graphic permits the use of simple perceptual operations to be substituted for more complex cognitive ones (e.g., visual comparison instead of mental computation).

Thus, there is a gap between the output of research systems that perform presentation planning by reasoning about communicative goals and acts, and systems that design data graphic by reasoning about tasks and the graphic techniques that support them. To bridge this gap, we propose translating communicative goals into *conceptual tasks*, so that if the users carry out those tasks on a selected set of data, they will achieve the communicative goals.

2. AN EXAMPLE
Consider a summary of an airline delivery schedule, whose communicative goal is to make the reader believe that *the destination airport with the most total tons of bulk and equipment cargo is Miami*. To achieve this goal in graphics, first we need to select data that express the fact. Clearly, the assertion is about the destination attribute of a discourse entity of type cargo identified by conditions on two of its attributes: cargo type (bulk and equipment) and quantity (the maximum total quantity among the cargo arriving at the different ports). This characterization of the entity can be expressed by data at different levels of aggregation. For example, we might use the individual cargo orders (containers of bulk cargo and pieces of equipment) and focus on the three relevant attributes – cargo type, quantity, and destination. Or, we might aggregate the cargo orders by destination and cargo type (i.e., consider all cargo orders of the same type that go to the same destination as one object, and focus on its summary attributes *common cargo type*, *common destination*, and *total quantity*). Notice that aggregating just by cargo type cannot express the fact in the goal because it does not allow computing the total quantity to a given destination. Hence, we need to spell out the rules by which expressive data sets can be selected for any given goal.

Let us assume that we have chosen the common cargo types, common destinations, and total quantities of all cargo aggregated by type and destination. Different graphics can portray this data such as Table 1, the stacked bar chart in Figure 1, or the two aligned charts in Figure 2. Table 1 expressed the data but to make the intended conclusion the viewer will have to scan the rows of the whole table, sum the two numbers in each row, and keep track of the row with the maximum sum. Therefore, Table 1 is very ineffective in realizing the sample goal. On the other hand, the message immediately jumps out in Figure 1. One instantaneously spots the longest stacked bar and looks up the name of the city associated with it. The stacked bar technique allows perceptual summing and parallel comparison vs. the slow cognitive computation and serial comparison in Table 1. Hence Figure 1 is not only expressive but also highly effective in achieving the goal. Finally, Figure 2 does not express the data at all even though it shows all attributes. The table shows destination ports, and the chart plots quantities versus cargo type. Since the user cannot associate the tonnage of cargo with their destinations, s/he will not be able to compute the total quantity arriving at each port.

* This work was done while both authors were in the Robotics Institute, Carnegie Mellon University.

The three graphics demonstrate that just selecting expressive data is not enough. For the graphic designer to exclude inexpressive graphics such as Figure 2, and to prefer expressive and effective graphics such as Figure 1 to expressive but ineffective ones such as Table 1, it needs to know what tasks the user will perform with the graphic. The tasks are the second piece of information (in addition to the data) that needs to be inferred from the communicative goal and supplied to the graphic designer so that it can select expressive and effective graphical techniques.

Table 1. Cargo aggregated by destination and type

Destination	Bulk (tons)	Equipment (tons)
Boston	0	115
Dallas	85	0
Denver	0	25
Houston	80	0
Miami	100	90
New York	110	0
Phoenix	0	45
Portland	75	0
San Diego	100	30
Seattle	80	15

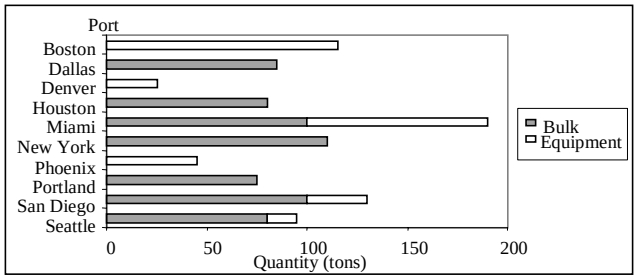

Figure 1. Cargo aggregated by destination and type

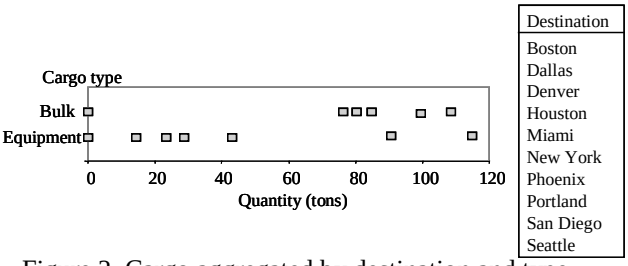

Figure 2. Cargo aggregated by destination and type

Suppose now that the summary should also convey the following fact: *a total of 4 pieces of equipment arrive at Miami*. This goal could have been achieved had we selected the individual cargo orders and the technique in Figure 1 (stacking the quantities of all cargo orders going to the same destination). The resulting graphic (Figure 3) not only achieves the first goal with the same effectiveness as Figure 1 but also achieves the new goal - one can easily count the four white bars in the longest stacked bar. The joint realization of more than one communicative goal in one graphic is called *aggregation* (not to be confused with data aggregation).

The benefit of aggregation is two-fold. First, achieving several goals in one graphic reduces the total *overhead*, which is the initial effort that the viewer makes to understand the presentation techniques. For example, to understand Figure 1, one must look at the caption, the axes, and the key to see what kinds of information is presented and how. Only then can the user make conclusions about the domain based on graphical relationships. When several acts share their overhead, the total interpretation is more efficient.

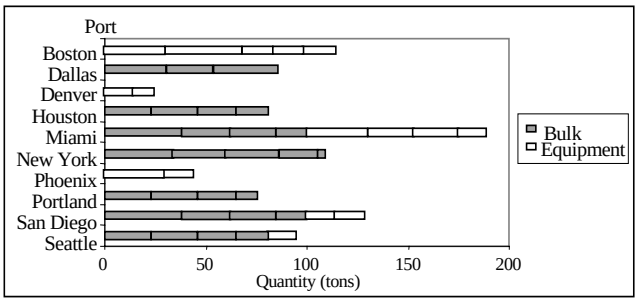

Figure 3. Bulk and equipment cargo orders

A second, more important, benefit of aggregation is the increased *coherence*. Coherent presentations provide facilities that help the user make the logical connection between different parts of a message. For example, in Figure 3, the longest stacked bar serves as an anchor for the two communicative acts realized by the graphic. The first goal results in the user recognizing that the longest stacked bar corresponds to the total cargo arriving to Miami. Then, to achieve the second goal, all the user needs to do is count the white pieces on that same stacked bar.

In this paper, we propose three conditions for aggregation: *homogeneous* goals, goals with *common identification*, and goals with *chained identification*.

This section demonstrated the following phenomena:

- Graphics present data about domain objects using graphical relationships.
- Users perform perceptual and cognitive tasks to achieve communicative goals in graphics.
- For a given communicative goal, some levels of data aggregation are expressive while others are not.
- For a given communicative goal and an expressive data set, some graphical techniques are expressive while others are not.
- Some techniques are more effective than others.
- The expressiveness and the effectiveness of the graphical techniques depend on the tasks that the user needs to perform with the graphic.
- The realization of more than one communicative goal in one graphic can reduce the overhead and increase the coherence of the presentation.

3. COMMUNICATIVE GOALS AND CONCEPTUAL TASKS

In this section, we look at the problem of mapping communicative goals to conceptual tasks from an architectural point of view. In particular, we show where it fits within the broader picture of communicating information and introduce the languages in which goals and tasks are expressed. We have studied the problem at hand in the context of AutoBrief, a multimedia (natural language and information graphics) generation system [5]. AutoBrief employs communicative planning and two media generators to produce a multimedia presentation that achieves a given high-level communicative goal (Figure 4). The communicative planner applies media independent presentation strategies to decompose the high-level goal into primitive communicative acts of type (Assert <proposition>) that achieve simple subgoals of type (Believe User <proposition>). After planning, media allocation rules assign the acts (and the subgoals they achieve) to text and graphics. In general, communicating complex quantitative relations or numerous homogeneous facts is more effective in graphics than in text. The

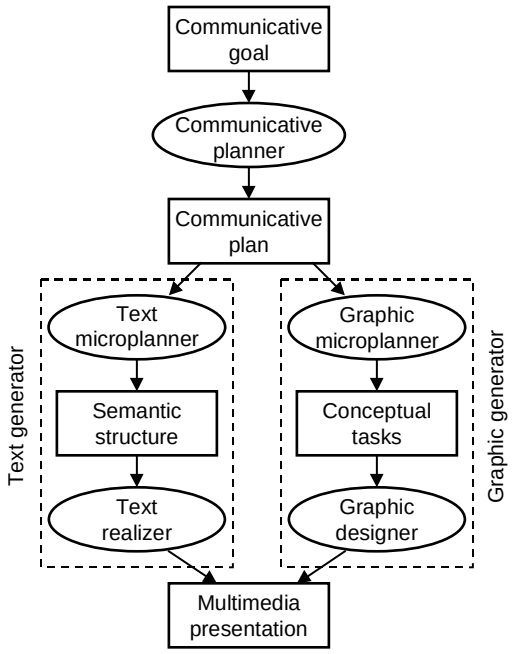

Figure 4. AutoBrief's architecture

two media generators produce English sentences and graphics, which are merged into a unified multimedia presentation.

The graphic generator consists of two modules: a graphic microplanner and a graphic designer. This paper provides the theoretical foundation of the microplanner, whose function is to translate the primitive communicative goals expressed in a logic-based language into conceptual tasks that operate on concrete data sets. The output of the microplanner comprises a design request acceptable by an automated graphic designer such as SAGE [12] or BOZ [3]. The designer then selects graphical techniques that support the conceptual tasks.

To define the knowledge that guides the microplanner, we need to be more specific about what goals and tasks actually are and how they are expressed. Communicative goals represent the message that needs to be conveyed to the user. The content of a goal, its proposition, is expressed in a logic-based language [13, 4]. For example, **sample-goal** below expresses the assertion from section 2 in such a language. As a reminder, this assertion reads *the destination airport with the most total tons of bulk and equipment cargo is Miami*.

Sample-goal:

```
MAIN PREDICATE (dest-c MAX-QUANT-CARGO Miami)
  MAX-QUANT-CARGO: the subset of ALL-CARGO-AGGRS
      whose total quantity is MAX-QUANTITY
  MAX-QUANTITY: the maximum of ALL-QUANTITIES
  ALL-QUANTITIES: the set of the total quantities
      of ALL-CARGO-AGGRS
  ALL-CARGO-AGGRS: all sets of cargo objects of
      type bulk or equipment that go to the same
      destination
```

This content expression specifies the main predication and the discourse entities using logical constructs such as quantifiers, domain types (e.g., cargo), domain predicates (e.g., dest-c and total-quantity), and general predicates (e.g., the maximum of a set). Each discourse entity is defined in a separate entry after the main predicate. Some of the entities (e.g., `ALL-CARGO-AGGRS`) define sets by imposing conditions on their members.

Conceptual tasks, on the other hand, prescribe operations that users perform on data objects to extract information about the domain. Using tasks as an intermediate representation between communicative planning and graphic design is justified by studies in cognitive psychology, which show that humans interpret graphics by performing simple perceptual and cognitive tasks [6, 10]. Some examples of such tasks are perceiving visual predicates (e.g., a bar is much longer than the rest), associating visual predicates with data objects (e.g., the longest bar with the airport receiving the most cargo), and inferring propositions about those objects (e.g., Miami is the port that receives the most cargo). Hence, communicative goals that we achieve in text by asserting facts are realized in information graphics by enabling the user to perform certain operations. Modeling those operations at the data level as conceptual tasks provides the graphic designer with important information to support the selection of graphical techniques.

We propose that the tasks be composed of three groups of primitives: value-accessing, entity-manipulating, and organization tasks. The value-accessing tasks produce values in one of three ways: by interpreting a constant (the `VALUE` task), by accessing the value of an attribute (the `ATTRIBUTE` task), or by computing a value (the `COMPUTE` task). The entity-manipulating tasks are SEARCH (for activating an object or a set of objects), `LOOKUP` (for evaluating attributes of objects), and `COMPARE` (for evaluating the relationship between attributes of objects). The organization tasks are `SEQUENCE` (establishes sequential execution of its subtasks where the output of a subtask serves as input to the next subtask), `DISJOINT` (marks independent execution of its subtasks) and `CONJOINT` (the subtasks jointly produce a common output).

Sample-task:

```
1.SEQUENCE
  1.1.DISJOINT
    1.1.1.SEARCH for {all-bulk-cargo}
                 in {all-cargo} by
          (= (cargo-type-c all-bulk-cargo) bulk))
    1.1.2.SEARCH for {all-eqpt-cargo}
                 in {all-cargo} by
          (= (cargo-type-c all-eqpt-cargo) eqpt))
  1.2.SEARCH for {(bulk-cargo eqpt-cargo)} in
             ({all-bulk-cargo} {all-eqpt-cargo})
      by (= (dest-c bulk-cargo)
            (dest-c eqpt-cargo)))
  1.3.SEARCH for (max-bulk-cargo max-eqpt-cargo)
             in {(bulk-cargo eqpt-cargo)} by
      (= (SUM (total-quant max-bulk-cargo)
              (total-quant max-eqpt-cargo))
         (MAX
          {(SUM (total-quant bulk-cargo)
                (total-quant eqpt-cargo))}))
  1.4.LOOKUP (dest-c max-bulk-cargo) Miami
```

Sample-task is a `SEQUENCE` of four subtasks (1.1-1.4) derived from **sample-goal** and realized in Figure 1. For clarity, while explaining the tasks we will refer to Figure 1. The tasks operate on a set of data objects, which we call *scope* of the tasks (and of the graphics that realize them). The scope of **sample-task**, `{all-cargo}`, is the set of all aggregates of bulk and equipment cargo whose members have the same destination and cargo type. `SEARCH` tasks 1.1.1 and 1.1.2 produce two subsets of `{all-cargo}`: those of type bulk (the gray bars in Figure 1) and of type equipment (the white bars). Subtask 1.2 identifies the pairs of bulk

and equipment aggregates with common destination (each pair is represented by one stacked bar). SEARCH task 1.3 identifies the pair of aggregates whose total quantity is the maximum among all pairs activated in task 1.2. The longest stacked bar expresses this pair. Finally, the LOOKUP task associates the destination of max-bulk-cargo (which is the same as the destination of max-eqpt-cargo) with Miami. In Figure 1, the ports are looked up as labels on the y-axis.

The main differences between the content descriptions of communicative goals and task specifications can be summarized as follows:

- The content language describes discourse entities by their properties while the task language expresses them by concrete data sets from an underlying database.
- The content language expressions are declarative, while the task language expressions are mostly procedural.

The mapping of communicative goals to conceptual tasks is presented in the next three sections by the following dimensions:

- Selecting a data set (scope) at an appropriate level of aggregation to express the entities involved in the goal.
- SEARCH tasks to activate the relevant objects within the selected scope.
- LOOKUP or COMPARE tasks to realize the main predicate.
- SEQUENCE, DISJOINT or CONJOINT relations to convey dependencies between the entity manipulating tasks to the graphic designer.
- Aggregating goals with certain relations between them and conflating their tasks into a single task.

We will illustrate the mapping rules using four goals (given below) supporting the hypothesis that insufficient port capacity at Miami is the possible cause for all late cargo arriving at that port. The goals are part of an explanation strategy based on the following pattern in the data: a port works at capacity level (communicative goals **1** and **2**), some cargo arrives late (goal **3**), and the late cargo arrives after the dates when the port works at full capacity (goal **4**).

Goal 1. On 6.03.1998[1] the usage of Miami equals its capacity.
Goal 2. On 6.04.1998 the usage of Miami equals its capacity.
Goal 3. About 200 tons of late cargo arrives at Miami.
Goal 4. The dates of arrival of the late cargo are after the dates when the usage of Miami reaches its capacity.

4. EXPRESSING PRIMITIVE DISCOURSE ENTITIES

In this section, we consider the expression of primitive entities, whereas the following section deals with sets. A primitive discourse entity is described in the communicative goal by the definitive quantifier ("the"), a primitive type (not a set), and a conjunction of conditions. Such entities are expressed by data objects in the domain database that satisfy the conditions imposed on the corresponding entity. Goal **1** of the sample plan illustrates this case. Discourse entities MIAMI-CAP-3 and MIAMI-USE-3 are of types port capacity and port usage, respectively, and are represented by the data objects retrieved from the database using their entity descriptions as queries. The scope of the tasks (and the graphic) must at minimum include those two objects. For goal **1**, the scope rule allows at least four alternatives: (**s1**) the capacity and usage of Miami on date 3 (two objects); (**s2**) all capacity and usage on date 3 (the number of objects is two times the number of ports); (**s3**) all capacity and usage of Miami (two times the number of days in the schedule); (**s4**) all capacity and usage (two times the product of ports and days in the schedule).

Goal 1:
```
MAIN PREDICATE: (= cap-throughput
                   used-throughput)
  cap-throughput: the amount of MIAMI-CAP-3
  used-throughput: the amount of MIAMI-USE-3
  MIAMI-CAP-3: the capacity of Miami on day 3
  MIAMI-USE-3: the usage of Miami on day 3
```

Three additional criteria with effectiveness flavor guide the selection of scope: (**e1**) to be as small as possible, (**e2**) to be described by a short logical form, and (**e3**) to promote aggregation. Criterion **e1,** linked to the principle of cognitive economy and parsimony [11], simplifies the presentation and reduces the side effects (non-planned assertions). Criterion **e2** facilitates the external identification of the scope [1], i.e. the description of the scope in the caption of the graphic (e.g., "all port capacity on 6.3.1998"). Criterion **e3** reduces the overhead and increases the coherence of the graphic. Under these criteria, if there was just goal **1,** then alternative **s1** would be preferable. With goal **2**, however, alternative **s3** becomes more desirable because it enables aggregation (cf. section "Goal and task aggregation").

Once a scope is selected, SEARCH tasks are included for each entity to *activate* the relevant objects in user's mind. The SEARCH is done using conditions on all attributes that both identify the entity and vary across the scope. Attributes that do not vary cannot differentiate the relevant objects within the particular scope, and therefore there is no use for SEARCH tasks by such attributes. Thus in scope **s1**, no SEARCH tasks are necessary because both identifying attributes are invariant. In scope **s2**, the SEARCH tasks are by port, in scope **s3** by date, and in scope **s4** by both port and date. If more than one SEARCH task is to be used, such as in scope **s4**, a CONJOINT operator should group the SEARCH tasks to inform the graphic designer that they produce their results jointly.

The main predicate is mapped to a LOOKUP task, if it predicates the value of an attribute, or a COMPARE task, if it predicates an equivalence, ordering, or arithmetic relation. Those mappings follow directly from the definition of the two tasks. In goal **1**, the arithmetic relation between the throughput amounts requires a COMPARE task.

A SEARCH, LOOKUP, or COMPARE task *T* depends on a SEARCH task *S*, if *S* produces (activates) objects that are used in *T*. Such a dependency implies that S must be performed before *T*, which means that *S* and *T* must be grouped by a SEQUENCE operator. If a task *T* depends on two independent tasks S_1 and S_2, the independent tasks are grouped by a DISJOINT operator before being grouped with *T* by a SEQUENCE operator. An example of this rule is shown below in **Task 1.1**, where the COMPARE task depends on two independent SEARCH tasks. Likewise, if two mutually independent tasks T_1 and T_2 depend on task *S*, they are first grouped by a DISJOINT operator and then grouped with S by a SEQUENCE operator.

The rules for task synthesis applied to goal **1** yield task **1.1** when the scope consists of all Miami capacity and usage (**s3**), {MIAMI-DAILY-CAPS} and {MIAMI-DAILY-USES}, respectively.

Task 1.1:
```
1.SEQUENCE
```

[1] From now on, the dates will be denoted by their day component only.

```
1.1. DISJOINT
  1.1.1. SEARCH for MIAMI-CAP-3 in
                    {MIAMI-DAILY-CAPS} by
         (= (date MIAMI-CAP-3) 3)
  1.1.2. SEARCH for MIAMI-USE-3 in
                    {MIAMI-DAILY-USES} by
         (= (date MIAMI-USE-3) 3)
1.2. COMPARE (= (amount MIAMI-CAP-3)
                (amount MIAMI-USE-3))))
```

Task **1.1** is sufficient for the graphic designer to generate the graphic in Figure 5. The following perceptual and cognitive operations lead the user to believe the proposition of goal **1**. The caption and the color key let the user understand that the gray and the black lines represent daily usage and capacity of port Miami. At this point, the user is aware of what the scope of the graphic is. The labels on the axes indicate that *x*-positions encode dates and *y*-positions encode throughput. The *x*-position encoding date 3 serves as a reference for finding the two overlapping marks that represent the capacity and usage of Miami on date 3. This accomplishes the two `SEARCH` tasks. Since the marks have the same *y*-position, the user concludes that the activated usage and capacity objects have equal throughputs, which was the goal. This accomplishes the `COMPARE` task.

5. EXPRESSING SETS

The expression of sets by data objects is challenging but also very flexible, giving opportunities for designing highly effective graphics. It is challenging because we can think of a set at different levels of aggregation: as the collection of its members, as one whole aggregate, or as a partition by subaggregates. Therefore, when making decisions about how to express discourse entities of type set, we have to carefully analyze the communicative goals in order to pick levels of aggregation that support what is meant to be conveyed. Before proceeding with this analysis, we will introduce some concepts related to data aggregation.

5.1. Data Aggregates

In this paper, we consider only aggregates created by imposing equality conditions on attributes of their members; e.g., *all cargo of type bulk*, or *all late cargo of type equipment*. The attributes we impose conditions on are called *aggregation attributes*. One can view this type of aggregation as binning, where a bin is created for each combination of values of the aggregation attributes and each object is placed into the bin corresponding to the values of the aggregation attributes of the object. The bins then are the aggregates and the objects inside them are their members.

Aggregates have their own attributes derived from attributes of their members. For example, the attribute *total-quantity* of a cargo aggregate is computed by *totaling* the quantities of its members; *status-c* is the common status of all members (or undefined if two members have different status). Attribute *a'* of aggregate *A* is a *summary* attribute derived from *base* attribute *a* using operator *S* if its values are computed by expression (**1**):

$$\textbf{(1)} \qquad a'(A) = \underset{o \in extension(A)}{S} a(o)$$

Thus, attribute *total-quantity* is derived from base attribute *quantity* using summary operator `TOTAL`. Attribute *status-c* is derived from base attribute *status* using operator `COMMON-VALUE`. Other common operators are `MIN`, `MAX` and `MEAN`.

An aggregate *A* is partitioned by subaggregates if no two subaggregates have common members and the total membership of the subaggregates is the same as the membership of *A*. Re-aggregating

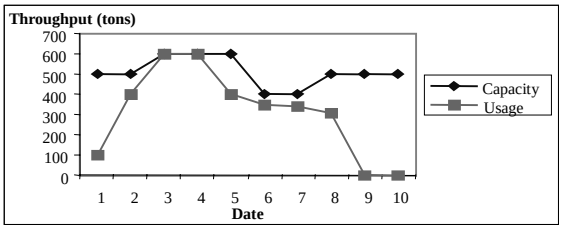

Figure 5. Daily capacity and usage for Miami

the members of an aggregate by some of their attributes partitions the aggregate. For example, re-aggregating the members of the aggregate of all cargo to Miami by their status produces two disjoint subaggregates, one for late and another for on-time cargo. Expression **(2)** lets us express assertions about a summary attribute (*a'*) of an aggregate (*A*) through assertions about its partition A_1, A_2, …, A_k (*S* is the derivational operator of *a'*):

$$\textbf{(2)} \qquad a'(A) = \underset{i=1}{\overset{k}{S}} a'(A_i)$$

5.2. Mapping rules

The mapping rules are based on how discourse entities of type set are defined and what is predicated about them.

A discourse entity of type set can be defined: (**d1**) by enumerating its members (extensionally), (**d2)** by imposing conditions on its members (intensionally), and (**d3)** as a subset in the partition of a superset through conditions on its summary attributes (cf. the definition of `MAX-QUANT-CARGO` as a subset of `ALL-CARGO-AGGRS` in **sample-goal**).

The predication about a discourse entity of type set can be: (**p1**) enumeration of its members; (**p2)** a predicate about all its members; and (**p3)** a summary attribute about the set.

An entity of type set can always be expressed by its members (**baseline** rule**).** However, this rule may not be the most effective one. If a summary attribute is asserted about a set expressed by its members, the user should compute the summary attribute to achieve the goal. Compare this to expressing the set by one aggregate object, of which the summary attribute is encoded directly by a graphical value. By the principles of effectiveness [7] and cognitive economy [11], we will seek the highest expressive level of aggregation to reduce the computation cost incurred by the user during graphic interpretation.

Expressing discourse entities of type set by data aggregates is subject to constraints on their level of aggregation. Those constraints are based on how the sets are defined and what is asserted about them.

a1. The set is defined or predicated about by enumerating its members (**d1** or **p1**). Since the intent is to focus on the members of the set, any aggregation would be inexpressive.

a2. The set is defined by equality conditions on attributes of its members (**d2**). Any aggregation should be done at least by the attributes defining the set. This rule ensures that the set will not be expressed as part of an aggregate and thus be unidentifiable as a separate graphical object(s). Recall how cargo aggregation just by cargo type in section 2 was inexpressive.

a3. The set is defined intensionally (**d2**) by non-equality conditions (e.g., "all the cargo that arrives before March 6"). A non-equality condition can be substituted with an equality condition on a derived Boolean attribute, which is true for all members that satisfy the non-equality condition and false

otherwise. In the example above, the cargo set can be redefined by the equality condition *arrives-before-March-6 is true*.

a4. The set is defined as a subset by conditions on some of its summary attributes (**d3**). Since the subset will be searched within the scope of the superset, both should have the same level of aggregation.

a5. A common predicate is asserted for all members of the set (**p2**). Any aggregation should be done at least by the predicated attribute. This will guarantee that the summary attribute derived using `COMMON-VALUE` will be defined for each member of the scope (see **c2** below).

a6. Predication about the summary attribute of a set (**p3**) does not impose any constraints on the level of aggregation except when the summary operator is `COMMON-VALUE`, which should be treated like asserting a common predicate for all members of the set (case **a5**).

The scope should contain all data objects at the selected level of aggregation that satisfy the conditions imposed on the discourse entity and is subject to the effectiveness criteria for scope selection (**e1**-**e3**) from section 4. Task synthesis is also similar to the case of primitive entities. The only difference is that the selected level of aggregation may result in using attributes in the tasks that are different from the attributes used in the goal. Those changes are summarized below:

c1. A discourse entity is defined intensionally and expressed by aggregates. The `SEARCH` tasks should use the summary attributes derived from the definition attributes by operator `COMMON-VALUE` (those attributes are guaranteed to be well defined because rule **a2** requires that all definition attributes are aggregation attributes as well).

c2. A common predicate is asserted about all members of an entity expressed by an aggregate. The `LOOKUP` or `COMPARE` task should use the summary attribute derived from the predicated attribute by operator `COMMON-VALUE` (this summary attribute is well defined due to rule **a5**).

c3. A summary attribute is predicated about a discourse entity expressed by its members. The summary attribute should be expressed by a `COMPUTE` task composed of the summary operator and an `ATTRIBUTE` task for the base attribute of the members. This mapping exploits formula **(1)** in section Data Aggregates.

c4. A summary attribute is predicated about a discourse entity expressed by a partition. The summary attribute should be expressed by a `COMPUTE` task composed of the summary operator and an `ATTRIBUTE` task for the summary attribute of the subaggregates. This mapping exploits formula **(2)** in section Data Aggregates.

5.3. Examples

Goal 3, shown below, predicates summary attribute total-quantity of entity `LATE-CARGO` defined intensionally by equality conditions on attributes status and destination. In Figures 6, 7, and 8, `LATE-CARGO` is depicted at three different levels of aggregation. In Figure 6 (realizing **task 3.1)**, it is expressed by one aggregate shown graphically as a bar. The aggregation is by the two attributes defining `LATE-CARGO` (rule **a2**). The scope consists of all late cargo aggregated by destination. Attribute destination was transformed into summary attribute dest-c (rule **c1**). Figure 7 (**task 3.2**) depicts `LATE-CARGO` by a partition of three subaggregates expressed by three vertical bars. The aggregation is by the two defining attrib-

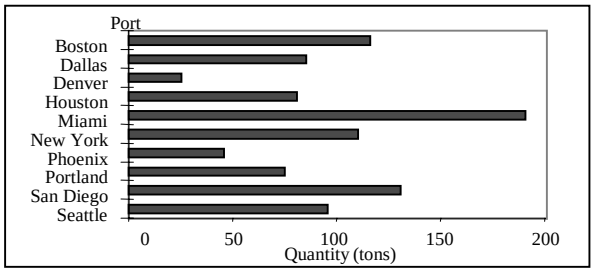

Figure 6. Late cargo aggregated by destination

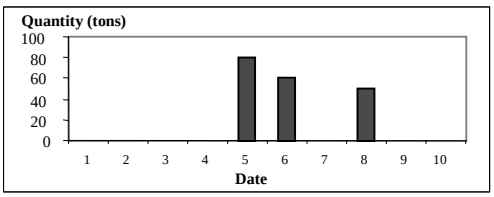

Figure 7. Late cargo to Miami aggregated by date

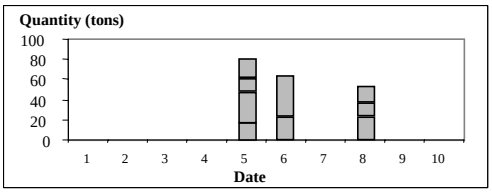

Figure 8. Late cargo orders to Miami

utes, status and destination, and by the partitioning attribute, arrival-date (rule **a2**). The use of arrival-date is justified in the next section on aggregation. To compute the total quantity, one has to perceptually sum the heights of the three bars. In Figure 8 (**task 3.3**), `LATE-CARGO` is expressed by individual cargo orders (**baseline** rule). There are no `SEARCH` tasks in **3.2** and **3.3** because none of the defining attributes varies. The total quantity of `LATE-CARGO` is expressed by `COMPUTE` tasks (`TOTAL`) over the total quantities in the partition of the set (task **3.2**, rule **c4**) and over the quantities of the members (task **3.3**, rule **c3**), respectively.

Goal 3:

```
MAIN PREDICATE: (total-quant LATE-CARGO ~200)
  LATE-CARGO: all late cargo to Miami
```

Task 3.1:

```
1. SEQUENCE
   1.1. SEARCH for MIAMI-CARGO in {CARGO-BY-DEST}
          by (= (dest-c MIAMI-CARGO) Miami)
   1.2. LOOKUP (total-quant MIAMI-CARGO) ~200
```

Task 3.2:

```
1. LOOKUP
    (TOTAL {(total-quant MIAMI-CARGO-BY-DATE)})
          ~200
```

Task 3.3:

```
1. LOOKUP
    (TOTAL {(quantity MIAMI-CARGO)}) ~200
```

Goal 4 asserts an ordering relation between the dates of all members of `AT-CAP-USE` (stands for "port usage at capacity level") and the arrival dates of `LATE-CARGO`. Figure 9 realizes this goal by supporting **task 4.1**. Entities `AT-CAP-USE` and `CAP` are expressed by the members of the sets (**baseline** rule). `LATE-CARGO` is expressed by aggregates by the two defining attributes status and destination (rule **a2**) and the predicating attribute arrival date

(since it applies to each element of the set, rule **a5**). Thus, the expression of LATE-CARGO combines two different constraints. Task **4.1** demonstrates the use of CONJOINT when an entity is searched by two attributes (date and amount). Notice also the substitution of attribute arrival-date by summary attribute arrival-date-c in the COMPARE task according to rule **c2**.

Goal 4:

```
MAIN PREDICATE: (< at-cap-date late-date)
  at-cap-date: the date of at-cap-use
  (AT-CAP-USE, CAP): all pairs of port-usage
      and port capacity of Miami such that the
      port usage and the port capacity components
      of each pair have equal dates and amounts
  late-date: the arrival date of LATE-CARGO
  LATE-CARGO: all late cargo to Miami
```

Task 4.1:

```
1. SEQUENCE
  1.1. CONJOINT
    1.1.1.  SEARCH for {(at-cap-use, cap)}
               in ({miami-uses}, {miami-caps}) by
            (= (date at-cap-use) (date cap))
    1.1.2.  SEARCH for {(at-cap-use, cap)}
               in ({miami-uses}, {miami-caps}) by
            (= (amount at-cap-use) (amount cap))
   1.2. COMPARE
        (< (date {at-cap-use})
           (arrival-date-c {late-cargo-by-date}))
```

6. GOAL AND TASK AGGREGATION

Early in the paper we showed how the realization of several goals in one graphic might reduce the overhead and increase the coherence of the presentation. Now we define three types of relations between goals that make their joint realization more effective than their realization in different graphics. The three aggregation rules based on those relations explain, at the task level, how Figure 9 was designed to achieve not only goal **4** but also goals **1**, **2** and **3**.

Goal **2** differs from goal **1** only by the dates of the capacity and usage entities (date 4 instead of 3). We regard goals **1** and **2** as *homogeneous* in the sense that they have isomorphic structures. If we select a scope that is expressive for both goals (e.g., scope **s3,** but not **s1** or **s2**), they would be mapped to the same types of tasks and realized by the same graphical techniques. The common graphic will be more effective than a separate one for each goal because the interpretation overhead will be shared by the two goals.

Goal **4** is related to goals **1** and **2** via the definition of entities AT-CAP-USE and CAP. These two entities are identified as a set of pairs by the condition that their amounts are equal, which is asserted about entities MIAMI-CAP-3 and MIAMI-USE-3 in goal **1** and their counterparts in the isomorphic goal **2**. We call this type of relation between communicative goals *chained identification*. Goals *A* and *B* are related by chained identification over entities e_1 and e_2 if e_2 is identified in *B* by a condition that is asserted about e_1 in *A*. The advantage of aggregating goals with chained identification over e_1 and e_2 is that the SEARCH task for e_2 derived from the chained condition in *B* is already satisfied by the tasks on e_1. Therefore, instead of creating a SEARCH task for e_2, a dependency must be established of any subtasks that use e_2 on the set of tasks that use e_1. Such a dependency is conveyed to the graphic designer by a SEQUENCE operator.

Goals **3** and **4** use the same definition of entity LATE-CARGO. We call this type of relation *common identification*. Goals *A* and *B* are related by common identification over entities e_1 and e_2 if e_1 and e_2 are identified by the same condition in *A* and *B*, respectively. The common identification enables a common search task for e_1 and e_2 and establishes dependencies of any other tasks derived from *A* and *B* that use e_1 and e_2 on the common search task.

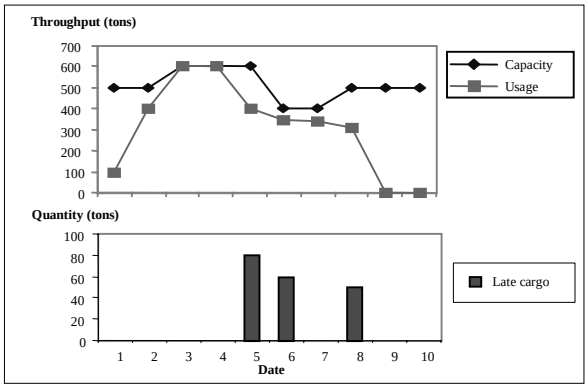

Figure 9. Daily capacity and usage for Miami and late cargo to Miami aggregated by date

The tasks for goals with chained or common identification over some entities can be aggregated only if those entities are expressed by the same objects and share the same scope (such as tasks **3.2** and **4.1** but not **3.1** and **4.1** or **3.3** and **4.1**). This rule justifies the choice of data aggregation by arrival date for expressing LATE-CARGO in tasks **3.2** even though arrival date is not used in goal **3**.

Aggregation based on the homogeneity of goals **1** and **2**, the chained identification relation of goals **1**, **2** and **4**, and the common identification relation of goals **3** and **4** conflates the tasks derived from them into a common set of tasks operating on a common scope. The graphic designer then generates Figure 9 as the best way to support the common tasks for all four goals.

The common tasks produce common graphic representations of the entities over which the goals are related. This is analogous to aggregation in natural language generation. For example, the sentences "Mary has blue eyes" and "John has blue eyes" can be aggregated into one sentence "Mary and John have blue eyes," where the shared concept of blue eyes is realized by a single noun phrase. Likewise, an entity used in two different goals and expressed by the same data object can be realized by a single graphical object. By expressing relations from multiple goals, such graphical objects make the presentation more economical and coherent.

7. PRIOR WORK

Prior work on automated graphic design focused on the selection of graphical techniques. Mackinlay's seminal work [7] set the foundation for computational synthesis of graphic presentations. He formulated the *expressiveness* and *effectiveness* criteria for graphical languages using properties of the data and properties of the human perceptual system. A graphical language is expressive with respect to a given set of facts if it encodes all those facts and only those facts. The effectiveness criterion, on the other hand, is rooted in the human ability to perceive some visual predicates better than others. It allows the comparison of different languages that express the same information, with respect to the efficiency of the interpretation of graphics by humans. Mackinlay's approach to graphic generation is based on the idea of composing graphical languages. In particular, he proposed three types of composition (double axes, single axis, and mark composition) and the condi-

tions for using them. He demonstrated those concepts in an automated presentation tool called APT.

Casner [3], Roth and Mattis [12], and Beshers and Feiner [2] pointed out the importance of user's tasks for selecting graphical techniques. These approaches posit composition of a set of task primitives, such as information search, lookup, comparison, and scan, that people engage in to accomplish more complex data exploration procedures. Their systems search for simple graphical techniques that transform primitives from complex cognitive operations to simple perceptual ones. Zhou and Feiner [14] used a different flavor of visual tasks, which abstract graphical techniques, rather than the exploratory behavior of the user.

Maybury [8] does realize some communicative goals in maps, but does not consider using the wide variety and complexity of graphics we are interested in. The limited set of graphical presentations in effect eliminates the need for automated design, making it possible to map communicative goals directly to graphics.

While this paper does not address the narrow problem of selecting graphical techniques, it advances the state-of-art of automated graphic design by proposing a model and rules for mapping communicative goals to conceptual tasks. This model reduces the problem of graphic realization of communicative plans to the problem of graphic design for tasks.

8. CONCLUSION

We proposed mapping communicative goals to conceptual tasks and three types of aggregation that lead to more effective graphics. This knowledge is essential for charting the search space that connects communicative planning with task-based graphic design and allows us to build systems that communicate information in graphics.

The mapping and aggregation rules presented in this paper are used in AutoBrief [5], a system that automatically summarizes transportation schedules in text in graphics[2]. In fact, AutoBiref designed the graphic in Figure 9 as part of the multimedia (text and graphics) explanation of a shortfall.

Unlike tasks in data exploration, the ones produced from communicative goals specify the results of the tasks. This enables the graphic designer to use attention-drawing devices such as highlighting a grapheme to attract user's attention to the relevant aspect of the graphic. For example, circling the two overlapping marks in Figure 9 would focus the user to the dates when the port is used at capacity level.

Our future work will address the computational aspects of generating graphics in context (i.e., how previously generated graphics may affect the design choices in subsequent presentations). Prior graphics set the context by the communicative goals they have achieved, possibly by some non-planned effects, by the data that have expressed the different discourse entities, and by the graphical techniques that have realized the tasks derived from the communicative goals. Each of these aspects will potentially influence the realization of new communicative goals in graphics.

[2] Some presentations generated by AutoBrief are available at http://www.cs.cmu.edu/~sage/ab-tour/start.html.

9. ACKNOWLEDGMENTS

This work was supported by a contract with DARPA, contract DAA-1593K0005. The authors would like to thank Mark Derthick, Joe Mattis and the anonymous reviewers for their comments on the paper. Many of the concepts crystallized from discussions with the other members of the AutoBrief group - Johanna Moore, Nancy Green, and Giuseppe Carenini.

10. REFERENCES

[1] Bertin, J. 1983. *Semiology of Graphics: Diagrams, Networks, Maps.* Madison, WI: The University of Wisconsin Press.

[2] Beshers C., and Feiner, S. 1993. AutoVisual: Rule-Based Design of Interactive Multivariate Visualizations. *IEEE Computer Graphics and Applications*, 41-49.

[3] Casner, S.M. 1991. A Task-Analytic Approach to the Automated Design of Information Graphic Presentations. *ACM Transactions on Graphics*, 10(2), 111-151.

[4] Green, N., Carenini, G., Kerpedjiev, S., Roth, S. F., and Moore, J. D. 1998. A Media-Independent Content Language for Integrated Text and Graphics Generation. Proc. *Workshop on Content Visualization and Intermedia Representations* (CVIR'98). Montreal, Canada.

[5] Kerpedjiev, S., Carenini, G., Roth, S. F., and Moore, J. D. 1997. AutoBrief: a multimedia presentation system for assisting data analysis. *Computer Standards and Interfaces*, 18, 583-593.

[6] Lohse, G. 1993. A Cognitive Model of Understanding Graphical Perception. *Human-Computer Interaction*, 8, 353-388.

[7] Mackinlay, J. 1986. Automating the Design of Graphical Presentations of Relational Information. *ACM Transactions on Graphics*, 5(2), 110-141.

[8] Maybury, M. T. 1991. Planning Multimedia Explanations Using Communicative Acts. In *Proc. AAAI-91*, 61-66.

[9] Moore J. D. 1995. *Participating in Explanatory Dialogues.* Cambridge, Massachusetts: MIT Press.

[10] Pinker, S. A 1990. Theory of Graph Comprehension. *In Artificial Intelligence and the Future of Testing*. Ed. R. Freedle, Hillsdale, N.J., Laurence Erlbaum Associates, 73-126.

[11] Rescher, N. 1989. Cognitive Economy: The Economic Dimension of the Theory of Knowledge. University of Pittsburgh Press.

[12] Roth, S. F., and Mattis J. 1990. Data Characterization for Intelligent Graphics Presentation. *Proc. SIGCHI'90*, Seattle, WA, ACM, pp. 193-200.

[13] Webber, B. 1983. So what can we talk about now? In B. Grosz, K. S. Jones, and B. Webber, eds, *Readings in Natural Language Processing.* Morgan Kaufmann, Los Altos, CA.

[14] Zhou, M., and Feiner, S. 1998. Visual Task Charactererization for Automated Visual Discourse Synthesis. Proc. CHI-98, Los Angelos, CA, 392-399.