Conjecture proved

Explanation:
Theorem used: thBt81233Tho81

\[ \text{IF } \text{connected}(2), \text{Nodes}=2*2+1 \text{ and Minimum}>=2 \]
\[ \text{THEN } \text{Circumference}>=2*2 \]

\[ \text{Nodes}=2*2+1=5 \]

Therefore the theorem holds and the conjecture is proved.

Figure 8. Conjecture proved, example 6

4. The Architecture of the AGORA Object-oriented GUI

The AGORA GUI is an important feature of the system, because, as we have seen, it must fulfill the graphist requirements on the GCP. It is an object-oriented GUI, in the sense that it is constituted by objects that can be manipulated directly. These objects correspond to the interface agents, based on the PAC (Presentation-Abstraction-Control) [1] multiagent model. This model, inspired in the MVC (Model-View-Controller) [10] model, preserves the paradigm of separation between the abstraction or semantic model of the application and its presentation aspects. The PAC interface agents, graphically represented by ovals in Figure 3, may be seen as three simple processors, with a memory and operations on this memory that can change its state, for capturing and processing events. They are organized according to the three perspectives: - Presentation, defining the image of the system with respect to the user interactions. - Abstraction, defining the semantics or functionalities of the system, constituting in general the data model. An abstraction can correspond to several presentations. - Control, acting as a mediator to maintain the consistency between the Abstraction and the Presentation; these perspectives only communicate through their respective control, to preserve the separation paradigm. Moreover, the PAC agents can only communicate through their controls. These communications are represented by straight line segments (see Figure 3). PAC agents recursively structure the whole system architecture in a tree-like organization. The agent in the upper level (root) represents the whole system and its abstraction constitutes the system data model. The remaining levels contain composite or elementary interface agents. The hierarchy ends with widgets or elementary agents that cannot be further decomposed.

The GUI architecture, is shown in Figure 3 below. The root in Level 0, corresponds to the AGORA data model, the abstraction (A) perspective of the agent. It contains the system data model. Its control, showing the AGORA name in the oval, communicates also with AKB and the temporary results (not shown in the diagram), which represents the persistent data of the system. Notice that the presentation (P) is absent in this agent representing the whole system. In fact, the system GUI presentation, is expressed by the Interface Manager (IM) agent, located in Level 1 and all the agents of the subsequent levels. Level 2 is constituted by four agents representing the user-interface modules specified in Figure 1: the Knowledge Base Browser (KBB), the Conjecture Editor (CE), the Conjecture Browser (CB) and the on-line Help System (HS). Level 3 in the PAC hierarchy contains the sub-agents constituting the presentation of the respective agent. KBB_menu with the main functionalities, KBB_edition the browser edition area with the edition facilities. CE_menu, CE_edition, CE_toolbar. CB_menu and CB_edition. HS_menu and HS_edition. Some snapshots of the interface have been shown in Section 3 (see Figures 3-8).
The PAC model, used to express the architecture of the AGORA GUI, offers several advantages: - the PAC agents correspond directly to graphic interface objects, facilitating an object-oriented user-interface design and development - the interface is easy to extend adding or removing PAC agents in the hierarchy, or just the presentation of an agent. Only the controls have to be modified. - the usual loss of control of object-oriented systems is avoided by the hierarchical structure of the PAC agents. - the loosely coupled architecture facilitate collaborative work, in the sense that the interface is easy to develop, assigning different agents or group of agents, to different programmers teams; the integration of the agents can be performed by a still different team, responsible of modifying only the controls. - an early prototype of the interface can be quickly presented, without having defined the system data model, which can be constructed in parallel or even after the whole interface component. Some disadvantages of this model may be due to a bad selection of the granularity of the agents, causing too many levels in the PAC hierarchy.

The AGORA GUI has been implemented in C++, under a Sun/Unix\textsuperscript{6}/MWWT++ platform. The use of frameworks to implement the PAC agents is strongly recommended. In fact, some GUI toolkits, as for example Visual C++, have now ready-to-use MVC like objects. The PAC framework has to be customized to the target language GUI toolkit, according to a precise specification [14]. We have implemented this framework for Java and C++. Particularly, the MWWT++[15], has been used to implement the GUI of the present AGORA version. It consists of several libraries providing a set of C++ classes, based on the PAC model and the Motif\textsuperscript{7} widgets libraries, corresponding to high level interface objects. These libraries facilitate object-oriented programming, integrating nicely the graphical interface objects with C++ classes.

5. Conclusion
During the study of a conjecture, the graphist checks whether the conjecture can be categorized i.e. proved, rejected or left open from the existing specialized knowledge. In most cases, this is not possible, and the researcher must conduct ad hoc methods for obtaining the veracity of the conjecture, or construct counterexamples (set of families) for rejecting it. There are no known general methods for solving a conjecture, so is up to the graphist to discover ingenious ad hoc processes. Hence, the goal of this work, is not the presentation of an automatic conjecture proving tool, but instead, to show a tool that can be used in a preliminary stage work of the researcher, to categorize a conjecture. The potentiality of rejecting conjectures in AGORA increases, if a good collection of test examples are available. These examples may be digraph families and digraphs with an attached description (nodemap) of their values and known invariant relations. The examples may be important keys to reject or build new conjectures. The potentiality of proving conjectures may be increased, generalizing and automating more methods commonly used by the graphist.

\textsuperscript{6}Unix is a registered trademark of Unix System Laboratories
\textsuperscript{7}Motif is a registered trademark of Open Software Foundation.