UNITED STATES DISTRICT COURT
DISTRICT OF MASSACHUSETTS

RED BEND LTD. and RED BEND SOFTWARE INC., Plaintiffs, v. GOOGLE INC., Defendant.	CIVIL ACTION NO. 09-cv-11813
GOOGLE INC., Counterclaim-Plaintiff, v. RED BEND LTD. and RED BEND SOFTWARE INC., Counterclaim-Defendants.	

GOOGLE INC.'S CLAIM CONSTRUCTION BRIEF

TABLE OF CONTENTS

TABLE OF AUTHORITIES

Page(s)

CASES

STATUTES

OTHER AUTHORITIES

I. INTRODUCTION

Red Bend's U.S. Patent No. 6,546,552 ("the '552 patent") covers a specific approach to creating software patches that can be used to update software programs. Sharon Peleg, the named inventor of the '552 patent, set out to address the well known problem of overly large software patches. The technique that he eventually settled on involves a particular way of processing the new and old executable programs to make modified versions of each in which references become "invariant" if they changed only due to the deletion or insertion of program instructions. The references that become invariant in the modified new and old executable programs are then eliminated from the difference result. This, the '552 patent claims, makes the difference result smaller than in prior art.

Peleg did not invent *any and every* way of making difference results relatively more compact than those found in the prior art. Rather, Peleg's particular technique of making references invariant and then eliminating them *is* the invention—according to the '552 patent itself, the prosecution history, Red Bend's opening preliminary injunction paper, Red Bend's expert, Google's expert, and the even according to the testimony of the inventor himself.

The parties' disagreement about the scope of the '552 patent animates their disagreement over the meaning of five claim terms:

1. "invariant references";
2. "compact difference result";
3. "data table";
4. "executable program"; and
5. the related terms "modified old data table," "modified old program," "modified new data table, and "modified new program."

See Ex. A (chart summarizing disputed and stipulated claim constructions).[1] Their differing

[1] As noted in Exhibit A the parties agree on the proper construction of the following claim terms: "old executable program" and "said old program"; "old data table"; "reference"; and "reference

(Footnote Continued on Next Page.)

constructions present three broad issues:

- Do the claims require that references made invariant be eliminated from the difference result as described in the specification and prosecution history?
- Do all claims require an executable program given the applicants' amendment of the clams and arguments made in order to overcome the Examiner's rejection of the claims as invalid in light of the prior art?
- Must the modified old/new programs/data tables be a version of the original executable program/data table with the invariant references inserted, as described in the intrinsic record, or does the claim language extend to programs and data tables modified in ways not described in the specification or prosecution history?

As detailed below, Google's proposed constructions are consistent with the claims themselves, the specification and the prosecution history. Google respectfully urges the Court to adopt them.

II. STATEMENT OF FACTS

A. The Asserted Claims

Red Bend alleges that Google's Courgette software infringes claims 8-12, 21-25, 28-34, 42-46, 55-60, and 62-67 of the '552 patent. These 34 asserted claims fall into several categories pertaining to the generation of a "compact difference result" (independent claims 8, 21 42 and 55), its storage on a storage medium (dependent claims 11, 24, 28, 45, 58 and 62), its transmission (dependent claims 9, 10, 22, 23, 29, 30-34, 43, 44, 56, 57, 60, 63, and 64-67), and the use of the claimed "compact difference result" to update software (independent claims 12, 25, 46 and 59). All disputed claim terms appear in the independent claims.

Claim 8 is representative of the server-side[2] claims:

(Footnote Continued from Previous Page.)

entries." Unless otherwise indicated, all referenced exhibits are attached to the Declaration of Susan Baker Manning In Support of Google Inc.'s Claim Construction Brief.

[2] Although the claims do not require that the "compact difference result" be created or stored on a "server," the parties have used "server-side" as a convenient shorthand for the claims directed

(Footnote Continued on Next Page.)

> 8. A method for generating a compact difference result between an old executable program and a new executable program; each program including reference entries that contain reference that refer to other entries in the program; the method comprising the steps of:
>
> (a) generating a modified old program utilizing at least said old program;
>
> (b) generating a modified new program utilizing at least said new program, said modified old program and modified new program have at least the following characteristics:
>
> (i) substantially each reference in an entry in said old program that is different than corresponding entry in said new program due to delete/insert modifications that form part of the transition between said old program and new program are reflected as invariant references in the corresponding entries in said modified old and modified new programs;
>
> (c) generating said compact difference result utilizing at least said modified new program and modified old program.

'Ex. B at claim 8 (emphasis added to disputed claim terms). Claim 42 is identical except that every reference to an "executable program" or "program" in claim 8 is replaced by the term "data table" in claim 42. Similarly, the only difference between claims 21 and 55 is the change from "executable program" or "program" in claim 21 to "data table" in claim 55. System claims 21 and 55 correspond, respectively, to method claims 8 and 42 in that each claims a system capable of performing the corresponding method, and does so in language identical to the method claim. As Red Bend and its expert have previously conceded, all four independent server-side claims are substantively identical. *See, e.g.*, Memorandum in Support of Plaintiffs' Motion for a Preliminary Injunction at 9 (Dkt. 9); Declaration of Stephen A. Edwards in Support of Plaintiffs' Motion at ¶ 25 (Dkt. 9, Attachment No. 4).

(Footnote Continued from Previous Page.)

to the generation of the "compact difference result" (independent claims 8, 21 42 and 55). Similarly, and consistent with the terminology of the specification, they have used "client-side" as a shorthand for the claims directed to the use of the "compact difference result" to update software (independent claims 12, 25, 46 and 59). *See* Ex. B at 1:21-46 (discussing the updating of an old program "installed at a remote client site").

The independent client-side claims (claims 12, 25, 46 and 59) each pertain to the updating of an "old executable program" on the remote client computer using a previously-generated "compact difference result."

> 12. A method for performing an update in an old executable program so as to generate a new executable program; each program including reference entries that contain reference that refer to other entries in the program; the method comprising the steps of:
>
> (a) receiving data that includes a compact difference result; said compact difference result was generated utilizing a modified old program and a modified new program;
>
> (b) generating a modified old program utilizing at least said old program;
>
> (c) reconstituting a modified new program utilizing directly or indirectly at least said modified old program and said compact difference result; said modified old program and modified new program have at least the following characteristics:
>
> (i) substantially each reference in an entry in said old program that is different than corresponding entry in said new program due to delete/inset modifications that form part of the transition between said old program and new program are reflected as invariant references in the corresponding entries in said modified old and modified new programs;
>
> (d) reconstituting said new program utilizing directly or indirectly at least said compact difference result and said modified new program.

Ex. B at claim 12 (emphasis added). Like the server-side claims, the client-side claims are identical in most respects. Claims 12 and 46 are method claims, while claims 25 and 59 are parallel system claims. Claims 12 and 25 recite an "executable program," while claims 46 and 59 recite a "data table."

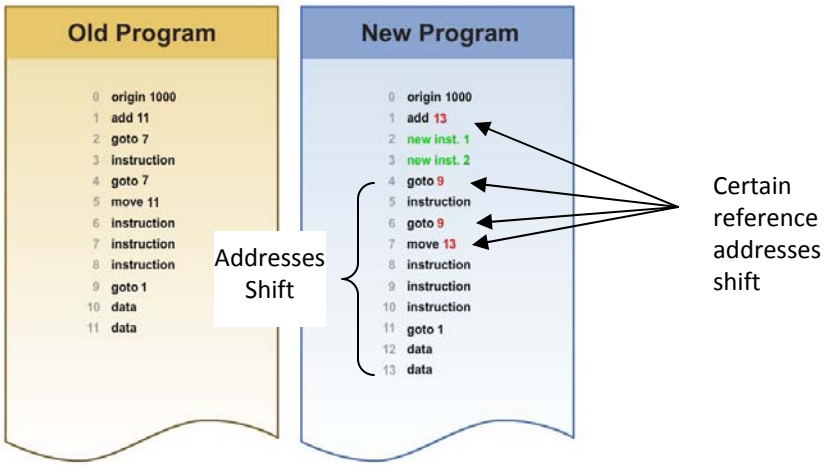
B. The Alleged Invention

1. The Disclosure of the '552 patent.

The '552 patent teaches a very specific way of reducing the size of difference results. It had been known for many years prior to the '552 patent that it was inefficient to update installed software by providing an entirely new version of the software every time there was any change,

no matter how small, vis-à-vis a previous software version. *See, e.g.*, Ex. B at 1:26-50; *see also* Ex. L at 1:32-37 ("Patching is an old technology, going back decades. Generally, patch files include a series of instructions specifying how a new version of a file can be assembled from snippets of data from an old version of the file, together with insertions of new data."). It was widely recognized that it was efficient to send a "patch" that contained just the changes between the new and old software versions, and use this to update the old program. Ex. B at 1:26-1:50. Long before the '552 patent, there were a number of difference generators available that could analyze new and old software versions, identify the differences between them, and create a difference result (or "diff") showing those differences. [redacted]

According to the '552 patent specification, prior techniques all contained a fundamental flaw in their handling of "reference entries"—that is, computer software instructions "which, by nature, specify a target address (reference) as an integral part of the command." Ex. B at 1:67-2:2. For example, in the instruction "move 11" at address 5 in the "old program," 11 is a reference entry. After new instructions (shown in green) are inserted, this becomes "move 13" in the "new program" below, with 13 being the new reference entry.



The problem with reference entries is that the simple insertion or deletion of instructions in a program, will result in a "ripple effect" of references that change only because other aspects of the program changed. Ex. B at 2:4-9; *see also* Ex. C at RedBend0000150; Ex. D at 14; [redacted]. This may result in an "inflated" difference result that contains not only the inserted or deleted section, but also includes all of the references entries altered due to the ripple effect. *See* Ex. B. at 1:59-2:19. For example, if new instructions are inserted after address 1 to create an updated program, ideally only the newly inserted instructions would have to be included in the difference result. However, what actually happens is that the address values associated with reference entries also change. In the prior art this resulted in their unnecessary inclusion in the difference result. *Id.* at 1:59-2:9. This is analogous to a redline of a document. If an author adds (or deletes) a single section heading to a document, all subsequent section headings are renumbered. Furthermore, any references to those renumbered section headings in the text also change even though only a single section heading has been added or deleted. According to the '552 patent, prior techniques would send the new heading and all changed references. *See, e.g.*, Ex. B at 3:36-46.

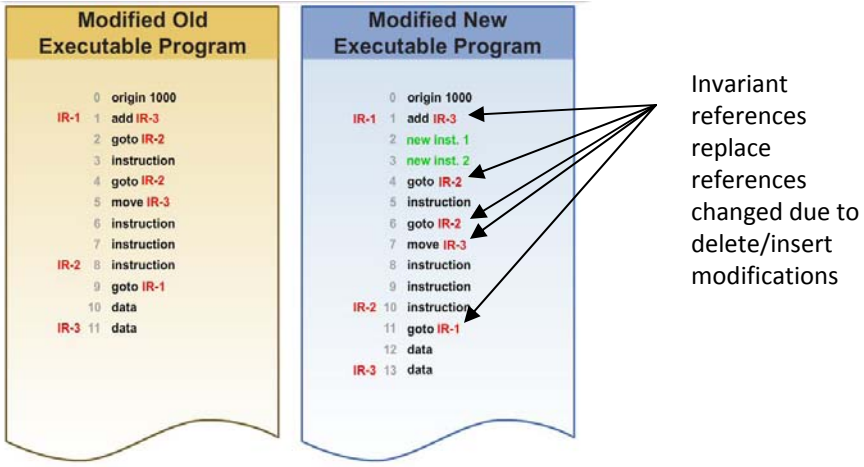
The '552 patent is directed to a particular solution for the problem of difference results that become "inflated" due to delete/insert modifications in a new program. *Id.* at 3:31-35. The '552 patent describes a specific technique for only sending what was deleted or inserted (*e.g.*, the new heading itself) in the patch, and excluding all references that change only due to the ripple effect.

> *The present invention* is based on the observation that the relatively large size of the difference result stems from the alterations of reference in reference entries as a result of other newly inserted entries (and/or entries that were deleted).
>
> On the basis of this observation, *the invention* aims at generating a modified old program and a modified new program, wherein the difference in references in corresponding entries in said new and old programs as explained above, will be reflected as invariant entries in the modified old and new programs. *The net effect is that the invariant reference entries (between the modified old program and the modified new program), will not appear in the difference result, thereby reducing its size as compared to a*

> *conventional difference result obtained by using hitherto known techniques.*

Id. at 3:36-46 (Summary of Invention) (emphasis added). Elsewhere in the specification, the applicant explained that shifts due to insertions are "neutralize[d]" with invariant references "with the obvious consequence that they are not reflected in the difference result, thus keeping the latter relatively compact." *Id*. at 9:66-10:15. [redacted]

The first step in eliminating the invariant references from a difference result, and thus making it "compact," is to identify those references that change only as the result of delete/insert modifications. Ex. B at 3:53-54. One then creates modified versions of the old and new programs in which substantially each such reference is replaced with an "invariant reference"—invariant in that it is the same in the modified old program and in the modified new program. *Id*. at 3:31-46 & 9:66-10:15; *see also* Ex. C at RedBend0000150. This is the only modification taught, and the old and new programs are otherwise unchanged from their original, executable form. Ex. B at 3:53-62, 4:8-19 & 12:11-28; Ex. C at RedBend0000151, RedBend0000173.



This replacement of references that change due to delete/insert modifications with invariant references makes them invisible to the difference generator. In essence, the invariant references trick the difference generator into thinking that certain references have not changed when in fact they have. For example, "move 11" in the old program became "move 13" in the

new program. In both of the modified versions of the old and new programs, these are replaced with "goto IR-3", with IR-3 being the invariant reference. Because the invariant references fool the difference generator into believing that the underlying references have not changed, references that change only as the result of delete/insert modifications are not included in the difference result. This elimination of references that change only as the result of delete/insert modifications makes the difference result smaller—or, in the language of the claims, makes it a "compact difference result." *See, e.g.*, Ex. B at claim 8, 3:36-46, 9:66-10:15.

2. The Prosecution History.

Red Bend filed the application that issued as the '552 patent on August 18, 1999. Ex. B On October 2, 2001, the Patent and Trademark Office issued an office action in which it rejected all claims as anticipated by each of two Japanese publications: JP 404242829A to Okuzumi, *et al.* and JP 05091550A to Kenji, *et al*. *See* Ex. C at RedBend0000128-135.; *see also* Ex. G (Okuzumi) & Ex. H (Kenji).

In response, the inventor amended claims 8, 12, 21 and 25, among others, to explicitly recite an "executable program," rather than merely a "program," and distinguished the invention on the basis of its manipulation of executable programs, as opposed to source or symbolic programs. Ex. C. at RedBend0000144-163. The applicant specifically argued that whereas Okuzumi concerns "source" programs—*i.e.*, programs as they exist in a human writable and readable programming language, or source code—"*the present invention* . . . defines an executable program." *Id*. at RedBend0000151 (emphasis added).[3] The applicant specifically

[3] An executable program causes a computer "to perform indicated tasks according to encoded instructions." *See* Merriam-Webster's Online Dictionary, http://www.merriam-webster.com/dictionary/executable (last visited July 15, 2010). The instructions of a program are executed directly by a computer's central processing unit. The files in which executable programs are stored are termed "binary files" or "binaries" because they encode instructions in the binary format used by the CPU, that is, 1s and 0s. In order to create an executable program, a programmer writes software code (sometimes called "source code") in a programming language or other symbolic form (such as assembly language), and then processes the code using a compiler. The compiler process may involve several intermediate steps, such as the creation of object files, but the ultimate output is the creation of a single executable program file containing

(Footnote Continued on Next Page.)

stated that, unlike Okuzumi, the claimed invention generates the difference result from executables:

> In extracting [a] diff between 2 versions of executable files . . . there is no source involved, and *neither statements, nor any textual or other symbolic representation of the program even exist*."

Ex. C at RedBend0000151 (emphasis added). The applicant also distinguished Okuzumi as relying on an auxiliary "order table" that "directs to source and not to [an] executable program." *Id*. at RedBend0000152.

Although the applicant's discussion of Okuzumi focused on claim 1 of the '552 patent, the applicant explicitly adopted each of these arguments as to all of the claims Red Bend new asserts. *See* Ex. C at RedBend0000149 ("Since the Examiner's detailed reasoning refers to Claim 1, the following, arguments focus on Claim 1, and will later apply to the other Claims, *mutatis mutandis*."); *id*. at RedBend0000152-53 (adopting "executable" arguments made as to claim 1 for claims 8 and 21). The applicant distinguished the independent "data table" claims 42 and 56 on the same grounds:

> Claims 35 to 68 are basically similar to claims 1 to 34, respectively, except for the fact that they recite data table instead of executable program. Data table is discussed on page 4, line 9 of the application *and do not embrace source code as in Okuzumi*. It is accordingly submitted that Claims … 42-44 [and] 55-57 are not anticipated by Okuzumi for the reasons discussed in detail above with reference to claims 1 to 3, 8-10. . . and 21-23.

Id. at RedBend0000154 (emphasis added). Likewise, the applicant distinguished all of the independent client-side claims the same grounds as the server-side claims. *Id*. at RedBend0000154-55.

The PTO issued a notice of allowance in August 2002. *Id*. at RedBend0000165-68. In

---

(Footnote Continued from Previous Page.)

1s and 0s, often termed a "binary," "executable," or "program". Source code and other human-readable forms are considered symbolic, as distinct from the binary code required of executables. Walker Decl. ¶ 20.

doing so, the Examiner noted that U.S. Patent No. 5,832,520 to Miller was "[t]he closest prior art reference of record currently" and that Miller teaches all of the elements other than "generating a modified new file and using the modified new file and the modified old file to generate the difference result." *Id*. at RedBend0000166; *see also* Ex. I (Miller). In response, Red Bend denied that "Miller creates modified old programs *in the sense of the invention*," as the Examiner had indicated. *Id*. at RedBend0000173-74 (emphasis added). Red Bend argued that Miller did not disclose a "modified old program" because Miller teaches the creation of "an index or hash table." *Id*. at RedBend0000173.[4] Red Bend described the "index or hash table" as "auxiliary data [] created in addition to the old file *and in contrast to the invention*[.]" *Id*. (emphasis added). Red Bend emphasized that, according to Miller, the creation of auxiliary data was not essential, whereas "the generation of modified old programs or modified old data table [sic] (depending on the aspects of the invention), is an obligatory step for the generation of the compact difference result." *Id*. at RedBend0000174. The Examiner disagreed with Red Bend's description of Miller. *Id*. at RedBend0000175-77. The '552 patent issued on April 8, 2003.

III. THE LAW OF CLAIM CONSTRUCTION

"It is a bedrock principle of patent law that the claims of a patent define the invention to which the patentee is entitled the right to exclude." *Phillips v. AWH Corp.*, 415 F.3d 1303, 1312 (Fed. Cir. 2005) (en banc). Claim construction is a question of law for the court alone. *Markman v. Westview Instruments, Inc.*, 52 F.3d 967, 970-71 (Fed. Cir. 1995). "The best source

---

[4] In computer science, a hash table is a data structure (*i.e.*, a way of storing and organizing data) that maps identifying values, known as keys (*e.g.*, a person's name) to their associated values (*e.g.*, their telephone number). One example would be a table in which the key is a person's name and hash table associates this key with the person's telephone number. A hash table is a type of index. Walker Decl. ¶ 26; *see also* Ex. J (WEBSTER'S NEW WORLD COMPUTER DICTIONARY) at 170 ("hash table": "a table of hash values that provides rapid access to data records . . . [t]he has function uniquely identifies each record, and the hash table includes pointers to each record."); Ex. K (MICROSOFT COMPUTER DICTIONARY) at 247 ("hash": "To be mapped to a numerical function . . . [h]ashing is used to convert an identifier or key . . . into a value for the location of the corresponding data in a structure, such as a table.").

for understanding a technical term is the specification from which it arose, informed, as needed, by the prosecution history." *Phillips*, 415 F.3d at 1315 (internal quotations omitted); *see also Vitronics Corp. v. Conceptronic, Inc*., 90 F.3d 1576, 1583 (Fed. Cir. 1996) (the specification "is the single best guide to the meaning of a disputed term"); *Markman*, 52 F.3d at 979 ("A patent's claims must be read in view of the specification, of which they are a part.").

Courts presume that claim terms mean what they say, as understood by a person of ordinary skill in the relevant art.[5] *Phillips*, 415 F.3d at 1313; *Johnson Worldwide Associates, Inc. v. Zebco Corp*., 175 F.3d 985, 989 (Fed. Cir. 1999). The presumption of ordinary meaning may be overcome where the applicant redefined or limited a claim term. One way this can be done is through specific definitions set forth in the specification, such as the Glossary definitions found in the '552 patent, or in the prosecution history. *Phillips*, 415 F.3d at 1316 (special definitions or intentional disclaimers of claim scope in the specification are "regarded as dispositive" of claim term meaning); *Intellicall v. Phonometrics*, 952 F.2d 1384, 1388 (Fed. Cir. 1992) ("an inventor [may] choose[] to be his own lexicographer").

An applicant can affect the scope of his or her claims through statements made during prosecution "redefining the term or by characterizing the invention in the intrinsic record using words or expressions of manifest exclusion or restriction, representing a clear disavowal of claim scope." *Teleflex, Inc. v. Ficosa North America Corp*., 299 F.3d 1313, 1327 (Fed. Cir. 2002). Such statements are binding. *Id.*; *Phillips*, 415 F.3d at 1317; *Southwall Techs., Inc. v. Cardinal IG Co*., 54 F.3d 1570, 1576 (Fed. Cir. 1995) ("The prosecution history limits the interpretation of claim terms so as to exclude any interpretation that was disclaimed during prosecution. . . . Claims may not be construed one way in order to obtain their allowance and in a different way

---

[5] Here, the parties agree that a person of ordinary skill in the art is someone with a bachelor of science degree in computer science (or an equivalent), who has approximately two years of software development experience, and some understanding of how source code is converted to machine instructions that can be executed by a computer. Walker Decl. ¶ 8.

against accused infringers.") (citations omitted). Global comments in the prosecution history "made to distinguish the applicants' 'claimed invention' from the prior art" limit all claims of patent. *Digital Biometrics, Inc. v. Identix, Inc*., 149 F.3d 1335, 1347 (Fed. Cir. 1998); *see also Honeywell Int'l, Inc. v. ITT Indus., Inc.*, 452 F.3d 1312, 1318 (Fed. Cir. 2006) (construing claim term to include feature characterized as "this invention" or "the present invention"); *SciMed Life Sys. v. Advanced Cardiovascular Sys.*, 242 F.3d 1337, 1343 (Fed. Cir. 2001) (construing term to include feature characterized as "the present invention"); *Pharmacia & Upjohn Co. v. Mylan Pharmaceuticals Inc*., 170 F.3d 1373, 1378 (Fed. Cir. 1999) ("key feature of the present invention" statement during prosecution surrendered scope).

IV. THE PROPER CONSTRUCTION OF THE DISPUTED TERMS

A. "invariant references" (all asserted claims)

Google Claim Construction	Red Bend Claim Construction
"invariant references": values made the same in the modified old and new programs (or data tables) for corresponding reference entries so that the reference addresses are excluded from the difference result.	**"invariant references"**: values made the same.

According to the language of every claim, invariant references are inserted into the modified old and modified new programs in place of reference addresses that have changed due to delete/insert modifications.

> said modified old program and modified new program have at least the following characteristics:
>
> (i) substantially each reference in an entry in said old program that is different than corresponding entry in said new program due to delete/inset modifications that form part of the transition between said old program and new program are reflected as <u>invariant references</u> in the corresponding entries in said modified old and modified new programs

See, e.g., Ex. B at claim 8(b)(i) (server-side) & claim 12(c)(i) (client-side). The replacement of these references with invariant references is essential; only by having a one-to-one, invariant relationship between the reference in the modified old program and the corresponding reference

in the modified new program can the invention to ensure that the invariant references entries are not included in the difference result. The applicant specifically states in the Summary of Invention that this is his solution—and his only solution—to the problem he sought to address. *Id.* at 3:36-46 ("The net effect is that the invariant reference entries (between the modified old program and the modified new program), *will not appear in the difference result, thereby reducing its size* as compared to a conventional difference result obtained by using hitherto known techniques.") (emphasis added).[6] The applicant reiterated elsewhere in the specification that "it is accordingly an object of *the invention*" that shifts due to deletions and insertions are "neutralized" with invariant references, "with the obvious consequence that *they are not reflected in the difference result*, thus keeping the latter relatively compact." *Id*. at 9:66-10:15 (emphasis added). The applicant made the same point in the prosecution history in order to overcome the rejection of the claims. Ex. C at RedBend0000150 ("In accordance with the present application such a need is reduced or eliminated, and what is required, is just to send the first few modified bytes. . . . The modification is effected in such a way that the references become "invariant" (see ['552 patent at 3:36-46]) thereby reducing the diff result and rendering it compact."). *Digital*, 149 F.3d at 1347 (characterizations of "the invention" are limiting); *Honeywell*, 452 F.3d at 1318; *SciMed*, 242 F.3d at 1343; *Pharmacia*, 170 F.3d at 1378.

[redacted]

[6] Red Bend and its expert relied this same portion of the specification as summarizing the invention in its preliminary injunction briefing. Memo. in Support of Plaintiffs' Motion for a Preliminary Injunction (Dkt. 9) at 6; Declaration of Stephen A. Edwards in Support of Plaintiffs' Motion (Dkt. 9, Attachment No. 4) at ¶¶ 16-17. Google's expert agreed. *See* Declaration of Martin G. Walker in Support of Google Inc.'s Opposition to Motion for Preliminary Injunction at ¶¶ 15-18. *See also C.R. Bard, Inc. v. U.S. Surgical Corp.*, 388 F.3d 858, 864 (Fed. Cir. 2004) (limiting statements describing the invention as a whole are "more likely to be found in certain sections of the specification, such as the Summary of Invention").

[redacted]

In light of the claims themselves, the key teachings of the specification, and the applicants description of the invention during prosecution, "invariant references" should be construed as "values made the same in the modified old and new programs (or data tables) for corresponding reference entries so that the reference addresses are excluded from the difference result." Walker Decl. ¶ 29.

B. "compact difference result" (all asserted claims)

Google Claim Construction	Red Bend Claim Construction
"compact difference result": a difference result in which references that have changed due to delete/insert modifications do not appear.	**"compact difference result"**: a difference result of a smaller size as compared to a conventional difference result (obtained by using techniques in existence prior to the invention of the patent-in-suit) in which the need to reflect changes to references due to delete/insert modifications is reduced or eliminated.

As discussed above, the key to the invention of the '552 patent is the elimination of references that change only due to delete/insert modifications from the claimed "compact difference result." Ex. B at 3:36-46 ("invariant reference entries … will not appear in the difference result") & 9:66-10:15 (shifts due to deletions and insertions "are not reflected in the difference result"); Ex. C at RedBend0000150 (difference result smaller due to elimination of invariant reference entries). The elimination of that unnecessary data is the very thing that makes the difference result "compact," and provides the "significantly smaller volumes of difference results between old programs and new programs" that the inventor was trying to achieve. Ex. B at 2:18-20. This is the only way of making difference results "compact"

disclosed in the '552 patent. [redacted] To construe the claims *not* to cover the very heart of Peleg's approach to difference results would unmoor the claims from the teachings of the patent and render them invalid. *See, e.g., Lizardtech, Inc. v. Earth Resource Mapping, Inc*., 424 F.3d 1336, 1344 (Fed. Cir. 2005) (claims construed to be broader than the single compression process disclosed in the specification were invalid for failure to meet the written description and enablement requirements of 35 U.S.C. § 112).

For these reasons, and the reasons discussed with respect to "invariant references," the term "compact difference result" should be construed as "a difference result in which references that have changed due to delete/insert modifications do not appear."[7] Walker Decl. ¶ 26.

C. "executable program" (claims 8, 12, 21 and 25)

Google Claim Construction	Red Bend Claim Construction
"executable program": a program comprising machine language instructions and corresponding bytes of data used by the program that are ready to be run on a computer, excluding source or other symbolic code.	**"executable program"**: a program comprising machine language instructions and corresponding bytes of data used by the program that are ready to be run on a computer.

In Glossary section of the '552 patent, the inventor specified that an "executable program" is "a loaded program in machine-memory" or "an executable-file." Ex. B at 2:61-65. The entries of an "executable program" are "individual machine instructions of the program or the individual data elements used by the program" and are therefore ready to be loaded and run by a computer. *Id.* These Glossary statements are consistent with the ordinary meaning of "executable program." Persons of ordinary skill in the art understand that an "executable

[7] The parties agree that a "difference result" is "data representative of the difference between an old program (or data table) and a new program (or data table) used to carry out an update of the old program." *See* Ex. A.

program" does not include source or other symbolic code that cannot be run on a computer. Walker Decl. ¶ 20. *See also* Ex. J (WEBSTER'S NEW WORLD COMPUTER DICTIONARY) at 134 ("executable program": "A program that is ready to run on a given computer. For a program to be executable, it first must be translated, usually by a compiler, into the machine language of a particular computer."); Ex. K (MICROSOFT COMPUTER DICTIONARY) at 200 ("executable program": "A program that can be run. [redacted]

[redacted]

[redacted]

The applicant's statements to the PTO confirm that the "executable program" cannot include source or other symbolic code. As noted above, the applicant amended claims 8, 12, 21 and 25 to recite an "executable program," and thus overcame the Examiner's rejection of all claims in light of the Okuzumi reference. Ex. C at RedBend0000160-163 (showing amendments); *id*. at RedBend0000151 ("*the present invention* . . . defines an executable program") (emphasis added); *id*. at RedBend0000151 ("In extracting [a] diff between 2 versions of executable files . . . there is no source involved, and neither statements, *nor any textual or other symbolic representation of the program even exist*.") (emphasis added); Ex. D at 11-12 ("the claimed techniques are intended to operate on files after references have been resolved to become numeric, as opposed to symbolic -- thereby permitting the techniques of the '552 Patent to be applied to executable files and data tables"). These characterizations of "the invention" are limiting. *Digital*, 149 F.3d at 1347; *Honeywell*, 452 F.3d at 1318; *SciMed*, 242 F.3d at 1343; *Pharmacia*, 170 F.3d at 1378.

The ordinary meaning of "executable program," the specification and the prosecution history all require that source or other symbolic code be excluded. The Court should therefore construe the claim term "executable program" as "a program comprising machine language instructions and corresponding bytes of data used by the program that are ready to be run on a computer, excluding source or other symbolic code."

D. "data table" (claims 42, 46, 55 and 59)

Google Claim Construction	Red Bend Claim Construction
"data table": a table of entries, each of which may have a different size. An executable program is one example of a data table. It cannot include source or other symbolic code.	**"data table":** a table of entries, where an entry is an addressable unit within the data table. Each entry may have a different size. An executable program is one example of a data table.

As the applicant stated in the Glossary, a "data table" is "a table of entries, each [of which] may have a different size." Ex. B at 2:33-34. Also according to the Glossary, "a data table can be an executable program." *Id*. at 2:61. During prosecution, however, the applicant successfully overcame the rejection of all claims by going further. In response to the Examiner's rejection of all claims, the applicant asserted not just that a "data table" *could be* an executable program, but that the claims were allowable because the claimed "data table" "do[es] not embrace source code as in Okuzumi." Ex. C at RedBend0000154 (specifically asserting that server-side "data table" claims 42, 55, and their dependents are allowable for the same reasons as the executable program claims); *id*. at RedBend0000154-55 (same for client-side "data table" claims 46, 59, and their dependents).[8] Thus, the applicant argued that—at least in the context of software programs relevant to this case—the claimed "data table" *must be* an executable program. *Teleflex*, 299 F.3d at 1327 (Fed. Cir. 2002); *Phillips*, 415 F.3d at 1317.

Put another way, both the Glossary definition and the applicant's disclaimer during prosecution must be given effect. The "data table" can be an executable program and, in the context of software, cannot include source or other symbolic code. The applicant specifically disclaimed a broader construction in order to obtain issuance of the patent. As with "executable

[8] Red Bend conceded early in the case that all of the asserted independent server-side claims are substantively identical, despite some variation in claim language. *See* Memorandum in Support of Plaintiffs' Motion for a Preliminary Injunction (Dkt. 9) at 9 & 11. Its expert's declaration testimony was that there was no material difference between "data table" and "executable program." Declaration of Stephen A. Edwards in Support of Plaintiffs' Motion at ¶ 25 (Dkt. 9, Attachment No. 4) ("I consider 'executable program' and 'data table' equivalent") & Ex. A thereto.

program," this is the key point of disagreement between the parties regarding "data table," and here too Red Bend is bound by the prosecution history. *Phillips*, 415 F.3d at 1317; *Southwall*, 54 F.3d at 1576. "Data table" is therefore properly construed as "a table of entries, each of which may have a different size. It cannot include source or other symbolic code. An executable program is one example of a data table."

E. "modified old program" &"modified new program" (claims 8, 12, 21 & 25); "modified old data table" & "modified new data table" (claims 42, 46, 55 & 59)

Claim Term	Google Claim Construction	Red Bend Claim Construction
Modified old program	A version of the actual program or data table in its original executable form, with certain portions replaced.	An interim result, such as tables or data structures, related to the old data table.
Modified new program		An interim result, such as tables or data structures, related to the old program.
Modified old data table		An interim result, such as tables or data structures, related to the new data table.
Modified new data table		An interim result, such as tables or data structures, related to the new program.

The modified old and new programs/data tables claimed in the '552 patent are what the claim language suggests they are: changed versions of the old and new programs/data tables. The modified old and new programs/data tables are created from the original old and new programs/data tables. The modification necessary for the invention to make a smaller difference result, and the only modification disclosed anywhere in the patent is the insertion of invariant references. Google's proposed definition acknowledges that the claims start with old and new executable programs/data tables and that the modified old and new programs/data tables are essentially those old and new executable programs/data tables, but with references that changed only due to delete/insert modifications replaced by invariant references.

The applicant emphasized to the Examiner that the modified old and new programs (or data tables) remain in executable format after modification. In all claims, the compact difference result is generated using the modified old program [or data table] and the modified new program

[or data table]. The applicant was explicit that the "diff" is "extract[ed] … between 2 versions of executable files" and "there is no source involved, and neither statements, nor any textual or other symbolic representation of the program even exist." Ex. C at RedBend0000151. Later in the same Response to Office Action, the applicant reiterated that "the present invention as defined in amended claims 1 concern executable program as compared to source programs in Okuzumi." *Id*. at RedBend0000152. The applicant made the point a third time, by distinguishing between the claimed invention and the "order table" of Okuzumi that "directs to source and not to [an] executable program." *Id*.; *see also id*. at RedBend0000153 (distinguishing claim 8 from Okuzumi because "[i]t concerns executable programs, applying processing steps to the references steps b(i)"); *id*. at RedBend0000154-55 (distinguishing all asserted claims from Okuzumi on the same grounds).

As to the replacement of certain portions of the old and new programs, the applicant made it clear that to "reflect[]" a reference as invariant is to *replace* it with an invariant reference. Ex. B at 10:47-50 ("the desired invariant references are accomplished by generating modified old and new programs wherein address references in entries are replaced by label marks"); [redacted]. *Cf*. Ex. B at 13:59-61 (new program is created on client side by "replac[ing]" invariant references with the actual address reference).[9]

Consistent with the specification, the applicant specifically asserted during prosecution that the claimed "modified old program [or data table]" and "modified new program [or data table]" from which the "compact difference result" is generated contain *replaced* references and

[9] The specification describes at length how to replace references (*i.e.*, addresses) within certain reference entries (*e.g.*, jump instructions) of an old program P1 and a new program P2. Ex. B at 12:11-28 ("step e" describing "replacing" address references associated within certain reference entries of a program with label marks) & Fig. 1. It also describes the modification process as one of replacing references associated with certain reference entries (*i.e.*, jump instructions) in a program or data table with an invariant reference, such as a label mark. *Id*. at Abstract ("replacing the reference of the entry by a distinct label mark, whereby a modified old [or new] program is generated."); 3:53-62; 4:8-19 (describing replacement of references in reference entries to create the modification).

not new, auxiliary data such as indexes or tables. During prosecution, the applicant argued that the Miller prior art patent cited by the Examiner does not "create modified old programs in the sense of the invention." Ex. C at RedBend0000173. The applicant was at pains to distinguish between the claimed invention and Miller's creation of an index or hash table. *Id*. "[A]ccording to Miller, auxiliary data is created in addition to the old file and in contrast to the invention[.]" *Id*. "Miller refers to generation of auxiliary index data structure. . . . Miller does not disclose the generation of a modified old file "replacing the reference of said entry by distinct label mark". *Id*. at RedBend0000174 (paraphrasing claim 1). As noted in the Abstract, "*[t]he method* includes the steps of scanning the old program and for each reference entry perform steps that include replacing the reference of the entry by a distinct label mark, whereby a modified old program is generated."[10] Ex. B at Abstract (emphasis added); *see also id.* at 3:31-46 & 9:66-10:15. Such characterizations of the invention are limiting. *Digital*, 149 F.3d at 1347; *Honeywell*, 452 F.3d at 1318; *SciMed*, 242 F.3d at 1343; *Pharmacia*, 170 F.3d at 1378.

Google's proposed construction of "modified old program [or data table" and "modified new program [or data table]" as "a version of the actual program or data table in its original executable form, with certain portions replaced" is consistent with both the claim language and the plain import of the specification, and also with the applicant's clear statements to the PTO during prosecution.

V. CONCLUSION

For all of these reasons, Google respectfully requests that the Court adopt its proposed constructions of the disputed claim terms.

---

[10] Claim 1, for example, recites the steps of "replacing the reference of said entry by a distinct label mark, whereby a modified old program is generated" and "replacing the reference of said entry by a distinct label mark, whereby a modified new program is generated." Although the asserted claims to not contain a specific requirement that the invariant reference be a "label mark," both the specification and the prosecution history show that the patentee's invention involved the replacement of references with invariant references.

Respectfully Submitted,

Dated: July 15, 2010

Google Inc.,
By its attorneys,

/s/ David M. Magee .
Jonathan M. Albano, Bar No. 013850
jonathan.albano@bingham.com,
David M. Magee, Bar No. 652399
david.magee@bingham.com
BINGHAM McCUTCHEN LLP
One Federal Street
Boston, MA 02110-1726
Telephone: 617.951.8000

Susan Baker Manning (*pro hac vice*)
susan.manning@bingham.com,
Robert C. Bertin (*pro hac vice*)
r.bertin@bingham.com
Elizabeth Austern (*pro hac vice*)
elizabeth.austern@bingham.com
BINGHAM McCUTCHEN LLP
2020 K Street, NW
Washington, D.C. 20006-1806
Telephone: 202.373.6000

William F. Abrams (*pro hac vice*)
william.abrams@bingham.com,
BINGHAM McCUTCHEN LLP
1900 University Avenue
East Palo Alto, CA 94303-2223
Telephone: 650.849.4400

Certificate of Service

I hereby certify that this document filed through the ECF system will be sent electronically to the registered participants as identified on the Notice of Electronic Filing (NEF) and paper copies will be sent to those indicated as non-registered participants, by federal express, on July 15, 2010.

/s/ David M. Magee
David M. Magee