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 RESPONSIVE CLAIM CONSTRUCTION BRIEF

TABLE OF CONTENTS

TABLE OF AUTHORITIES

Page(s)

CASES

STATUTES

I. INTRODUCTION

There are five disputed terms to be construed by this Court. Red Bend concedes that the specification and prosecution history informs the proper construction of some of them—but not all. Even where it recognized the relevance of the intrinsic evidence, it either applies it selectively (as in its "data table" definition) or misapplies it (as in its proposed constructions of "compact difference result" and the modified old/new program/data table limitations). As to, "invariant reference" and "executable program," Red Bend entirely disregards the teachings of the specification and prosecution history.

Through its proposed constructions, Red Bend would rewrite the '552 patent to cover, essentially, a method of generating a difference between an old and new executable program (or data table) such that at least one reference that has changed due to delete/insert modifications is represented in the difference result in a more compact way than prior to the filing of the patent. This is clearly contrary to the '552 patent. As shown below, Google's constructions are supported by intrinsic evidence and should be adopted.

II. THE PROPER CONSTRUCTION OF THE DISPUTED TERMS

A. "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.

Google's construction specifies that the claimed "compact difference result" is "a difference result in which references that have changed due to delete/insert modifications do not appear." That construction flows directly from the intrinsic evidence. Google Inc.'s Claim Construction Brief ("Google Br.") at 14-15 (citing and discussing Ex. B at 2:18-20, 3:36-46 & 9:66-10:15 and Ex. C at RedBend0000150) (Dkt. 93).

During the Preliminary Injunction process, Red Bend contended that any difference result smaller than one generated by the prior art would be the claimed "compact difference result." *See, e.g.*, Plaintiff's Reply in Support of Its Motion for a Preliminary Injunction ("PI Reply Br.") at 13 (Dkt. 58). Red Bend now concedes that, according to the specification and prosecution history, at least some of the of references that change due to delete/insert modifications must be eliminated from the claimed "compact difference result." Red Bend's Opening Claim Construction Brief ("Red Bend Br.") at 10 (discussing concession that "the need to reflect changes to references due to delete/insert modifications *is reduced* or eliminated") (Dkt. 96).

Red Bend's construction is nevertheless erroneous for at least two different reasons. First, its assertion that the number of relevant references that do not appear in the difference result need only be "reduced"—without any indication of by how much—is inconsistent with the "substantially each reference" limitation. Each asserted independent claim requires that:

> *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.

See, e.g., Ex. B at claim 8(b)(i) (emphasis added). The aim of the invention is the creation of a smaller difference result from which references made invariant have been eliminated. *Id*. at 3:36-46 & 9:66. Although Google believes that even under Red Bend's construction of "compact difference result" the claims require that substantially each reference that changes due to delete/insert modifications must be eliminated, Red Bend's proposed construction needlessly clouds the issue, and threatens to confuse the jury.

Second, Red Bend's construction is itself indefinite. A claim is indefinite, and therefore invalid, if it is "insolubly ambiguous" such that a person of ordinary skill in the art cannot determine the meets and bounds of the claim. *Honeywell Int'l, Inc. v. Int'l Trade Comm'n*, 341 F.3d 1332, 1338-39 (Fed. Cir. 2003); *Halliburton Energy Servs., Inc. v. M-I LLC*, 514 F.3d 1244,

1251 (Fed. Cir. 2008) ("Even if a claim term's definition can be reduced to words, the claim is still indefinite if a person of ordinary skill in the art cannot translate the definition into meaningfully precise claim scope."). Red Bend's construction is explicitly relative; it would define the "compact difference result" as being "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)." This assumes that the person of ordinary skill in the art would know how large a difference result might be obtained using any prior art difference generator on any given new and old executable program. The parties agree on the definition of a person of ordinary skill in the art. Google Br. at 11 n.5. This person need not have any specific knowledge of prior art difference result generators, much less the encyclopedic knowledge that Red Bend's construction would require. Red Bend's construction is erroneous for this reason as well. *See Amgen Inc. v. Hoechst Marion Roussel, Inc.*, 314 F.3d 1313, 1341 (Fed. Cir. 2003) (claim phrase that differentiated the claimed product relative to the prior art was indefinite).

Given its concession, Red Bend's only issue with Google's construction is based on a misreading of it. Red Bend urges that under Google's construction, Google "would likely argue" that all references that change due to delete/insert modifications must be eliminated from the difference result. Red Bend Br. at 10. However, Google's proposed construction of "compact difference result" does not address the separate "substantially each reference" limitation, and nothing in Google's proposed construction would or could read that limitation out of the claims.

If the Court were to construe the claimed "compact difference result" as "a difference result in which *substantially each* reference that has changed due to delete/insert modifications does not appear" the non-infringement analysis would be the same as under Google's construction above when each claim is read as a whole. While Google believes that appending the "substantially each" language to the "compact difference result" construction is unnecessary in light of the claim language, Google would not oppose such a construction if the Court believed that it further clarified the meaning of the claims.

B. "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.

Red Bend relies on the specification and prosecution history to define the claims to require that the references that change due to delete/insert modifications be made invariant so that they are "reduced or eliminated" from the claimed "compact difference result." Red Bend Br. at 9-11. When it comes to a similar construction of "invariant references," however, Red Bend accuses Google of improperly importing limitations from the specification. At bottom, Red Bend's criticisms of Google's proposed construction of "invariant references" contradict its own proffered constructions. To the extent that Red Bend argues that context is unnecessary, Red Bend is unpersuasive. Patent cases are notoriously complex, and the trier of fact is far more likely to be aided by the context than to be confused by it.

Red Bend's construction of "invariant references" as simply "values made the same" fails to situate the term in the claims. Without that framework, Red Bend's definition makes the scope of the claims less, not more, clear. For example, the invariant references are not *any* "values made the same," they are values for corresponding references entries. Ex. B at 3:36-46. Google's definition makes this plain, while Red Bend's does not.

Elsewhere, the claims make it clear that the references are made the same in the modified old and new programs/data tables—just as stated in Google's construction. *Id.* at 16:30-37. Red Bend does not dispute that the invariant references take the place of corresponding reference entries in the old and new executable programs/data tables, *id*., or that the insertion of the invariant references is what allows the invention of the '552 patent to fool the difference generator into excluding references that change only due to delete/insert modifications from the difference result, *id*. at 3:31-3:46 & 9:66-10:15. Google's construction is consistent with these

teachings.

Finally, Red Bend's argument that the '552 patent makes a distinction between "reference" and "address" is based on made-up definitions that are not only unsupported by the Glossary statements Red Bend cites, but directly contrary to them. In the Glossary, the patent states "[a] reference can be either an address or a number used to compute an address." *Id.* at 2:43-44. Google's construction is consistent with this definition.

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.

The only dispute as to "executable program" is whether it must exclude source or other symbolic code. The ordinary meaning of "executable program" is a program that is ready to be run, and it therefore cannot include source or other symbolic code. Google Br. at 15-16 (citing dictionaries). This is consistent with the patentee's description of an "executable program" in the Glossary, and the applicant's arguments that in the claimed invention of the '552 patent "there is no source involved, and neither statements, nor any textual or other symbolic representation of the program even exist." *Id*. *See also* Ex. D at 12 (Red Bend's recent statement in reexamination affirming that "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").

Google's construction is consistent with the preferred embodiment. In fact, requiring the method to start with executable programs/data tables *is* the preferred embodiment. Red Bend has admitted that the applicant's amendments and statements to the Examiner limited the claims to starting with an "executable program" that does not include source or other symbolic code. *See*

PI Reply Br. at 7; Reply Declaration of Stephen A. Edwards in Support of Plaintiff's Motion for a Preliminary Injunction ("Edwards PI Reply Decl.") at ¶ 9 (Dkt. 60). The specification describes only one method of altering an executable file: a process in which references are replaced by invariant references or labels, as was reiterated to the patent office during prosecution. Ex. B at 9:66-10:15 & 10:47-12:28; Ex. C at RedBend0000174. Red Bend now hedges, and suggests that so long as not all of the program is source or symbolic code, this could be the an "executable program" within the meaning of the claims. Red Bend Br. at 18. This is contrary to the intrinsic evidence. Likewise, Red Bend's contention that executable files contain symbolic code is directly contrary to its statements to the Examiner, and have no weight here. Ex. C at RedBend0000151.

Ironically, Red Bend's construction departs from its own expert's explanation that a "program" does not include source or other symbolic code. Edwards PI Decl. at ¶ 12 ("For a program to run on a computer, its source code must be . . . 'compiled,' into an executable program . . . [which] consists primarily of machine code instructions and references to other parts of the program.") In contrast, Google's proposed construction is consistent with Edwards' original explanation.

Red Bend's other critiques also fall short. For example, Red Bend's contention that executable files contain symbolic code is irrelevant (even if it were true) because the claim term is "executable program," not "executable file." Red Bend uses the terms "executable," executable file(s)," and "executable program" interchangeably, but they are not synonymous. *See, e.g.*, Ex. K at 200 (separately defining "executable" and "executable program").

Furthermore, Red Bend misquotes both experts, neither of whom have said that executable files "contain a 'relocation table' which is symbolic." Red Bend Br. at 17. Edwards, for example, merely parroted the '552 patent at 3:3-4 which recites "a relocation table *attached* to executable programs" (emphasis added). Although Dr. Edwards asserts that a *relocation table* contains symbolic code, he does not state that an *executable program* would contain symbolic

code. Edwards PI Reply Decl. ¶ 27. In fact, the patent distinguishes an executable program, which the parties agree comprises "machine language instructions and corresponding bytes of data used by the program that are ready to be run on a computer," from a relocation table. In other words, Red Bend's discussion of "executable file" and "relocation table" fails to distinguish Google's construction of "executable program" and the preferred embodiment.

Finally, Red Bend's characterization of the Okuzumi reference is inaccurate, and therefore its analysis of statements made during prosecution is unfounded. The applicant was forced to distinguish its alleged invention because Okuzumi describes key features. For example, much like the '552 patent, Okuzumi discloses processing to modify old and new programs, identification of delete/insert modifications, and changing references to invariant references in modified programs in order to produce a compact difference result. *See* Ex. G (Okuzumi) at ¶¶ 0018-22, 0025-26, 0044-45, 0052, 0058, and Figs. 6a, 6b, 8c & 8d; RedBend0005439-44; RedBend0005448-49. The key distinction was that Okuzumi uses source or other symbolic code and the process as claimed in the '552 patent does not.

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. It cannot include source or other symbolic code. An executable program is one example of a data table.	**"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.

It is undisputed that the Glossary's definition of "data table" informs the correct construction of the term, and both parties draw heavily on it. Google's proposed construction, unlike Red Bend's, also gives the prosecution history the weight required by Federal Circuit precedent. Like the claimed "executable program," the claimed "data table" cannot include source or symbolic code because Red Bend disavowed any broader claim scope in order to obtain issuance of the patent. *Teleflex, Inc. v. Ficosa North America Corp.*, 299 F.3d 1313, 1327 (Fed. Cir. 2002) ("clear statements of scope in the specification and prosecution history

determine[] the correct claim construction"); *Phillips*, 415 F.3d at 1317; *Southwall Techs., Inc. v. Cardinal IG Co.*, 54 F.3d 1570, 1576 (Fed. Cir. 1995) ("Claims may not be construed one way in order to obtain their allowance and in a different way against accused infringers.").

As described in Google's opening brief, Red Bend overcame the Examiner's rejection of the "data table" claims by arguing that, as defined in the Glossary, they do not include source code. Google Br. at 17-18. It specifically asserted that the claims were patentable over Okuzumi:

> In extracting diff between 2 versions of executable files as defined in amended Claim 1, **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). Red Bend acknowledges the statement in its brief, but argues that it pertains only to claim 1. *See* Red Bend Br. at 16. That is flatly wrong. Red Bend made precisely the same argument as to the data table claims:

> 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 [at Column 2, line 49 of the '552 patent] **and do not embrace source code as in Okuzumi**. It is accordingly submitted that Claims 39-41, 42-44, 48-50, 52-54, 55-57 are not anticipated by Okuzumi for the reasons discussed in detail above with reference to Claims 1 to 3, 8-10, 14-16, 18-20, and 21-23.

Ex. C at RedBend0000154 (emphasis added). Red Bend explicitly adopted the argument that the claimed programs and data tables do not include source or symbolic representations as to each and every asserted claim. Ex. C at RedBend0000151-55.

Dr. Edwards' claim that $P'_{1,}$ which he identifies as a modified data table, includes "symbolic code because the numbers in P1 are symbols" is misleading. Edwards PI Reply Decl. p. 7 & Declaration of Stephen A. Edwards in Support of Plaintiffs' Proposed Claim Construction ("Edwards Decl.") ¶ 10 (Dkt. 98). As explained on page 6 above, the specification describes a

process in which references in a file that contains no symbolic code are replaced by invariant references or labels. No other symbolic code is introduced in the described process. A data table input into and processed by the described method would have no symbolic code.

Red Bend's only other argument is that the Court should not give the prosecution history weight because the claims then would not read on the data table embodiments described in the specification. Red Bend Br. at 16. Regardless of the impact on the embodiments, Red Bend is bound by its own clear disavowal of claim scope. *Teleflex*, 299 F.3d at 1327; *Phillips*, 415 F.3d at 1317; *Southwall*, 54 F.3d at 1576. Although the Federal Circuit has recognized that constructions that exclude purported embodiments are to be scrutinized, they are appropriate "if the prosecution history compels such a result" as it does here. *North American Container, Inc. v. Plastipak Packaging, Inc*., 415 F.3d 1335, 1346 (Fed. Cir. 2005).

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.

As Red Bend acknowledges, the '552 patent's specification is the key source of information about what the claimed modified old program/data table and modified new program/data table actually are. The specification teaches, and the prosecution history requires, that they are: "a version of the actual program or data table in its original executable form, with certain portions replaced." *See* Google Br. at 18-20.

1. The modified old and new programs/data tables must be "*a version* of the actual program or data table *in its original executable form*"

Google's construction of this element adheres closely to the intrinsic evidence. The modified old and new programs/data tables must each be "a version" of the original old and new programs/data tables still in its "original executable form" because Red Bend asserted during prosecution 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; *see also* Google Br. at 19.

Red Bend misstates Google's construction in order to attack it, particularly isolating the phrase "in its original executable form" for criticism. Red Bend points out that in the embodiments described in the specification, the modified new and old programs/data tables are not in their "original executable form." Red Bend Br. at 24. But of course they are not. The specification describes at length how the modified new/old programs/data tables are changed so that invariant references replace the references that changed only due to the deletion or insertion of code. Ex. B at 9:67-10:15, 10:42-13:4. The claims require as much. *See, e.g., id*. at claim 8(b). That is why the modified old program is *a version* of the original executable program. There is no description of any working alternative in the specification or elsewhere.

Google has never contended that the modified new and old programs/data tables can themselves be executed prior to being "diffed."[1] The replaced invariant references do not actually refer to memory locations. A computer that tried to run, for example, the modified new program would come across an invariant reference, not know how to resolve it to an actual memory location, and simply crash. Red Bend's argument that the modified old and new programs/data tables described in the specification are not in their "original executable form" is of no moment because Google has never suggested that they are and its complete definition does not require them to be.

[1] Red Bend has previously acknowledged this. *See* Plaintiff's Reply in Support of its Motion for a Preliminary Injunction at 4-5 ("Google's position is *not* that the modified programs/data tables have to be 'executable'") (emphasis in original) (Dkt. 58).

The principle of claim differentiation does not aid Red Bend. First, its selective advocacy of the doctrine should be rejected. Red Bend long ago conceded that the '552 patent claims are redundant of one another. *See* Red Bend PI Memo. at 9-11; Edwards PI Decl. at ¶ 25. Second, claim differentiation principles inform the claim construction analysis, but are not dispositive. *Laitram Corp. v. Morehouse Industries, Inc*., 143 F.3d 1456, 1463 (Fed. Cir. 1998) ("although different claims should be presumed to cover different inventions, 'if a claim will bear only one interpretation, similarity [with another claim] will have to be tolerated.'"); *ATD Corp. v Lydall, Inc.*, 159 F.3d 534, 541 (Fed. Cir. 1998) ("The presumption that separate claims have different scope is a guide, not a rigid rule.") (internal quotations omitted).

Red Bend nevertheless urges that the modified old and new programs/data tables should not be construed as "a version of the actual program or data table *in its original executable form*, with certain portions replaced" because the modifier "executable" appears only in the preamble, not the body, of the claim. This is a non-sequitor. Red Bend does not deny that the modified old and new programs/data tables are versions of the old and new *executable* programs/data tables recited in the preamble. Google's definition simply recognizes that the modified versions of each remain in executable form except to the extent that certain references are replaced by invariant references. Ex. B. at claim 8, 3:36-46, 9:66-10:15.[2] This is consistent with the specification, which does not suggest any sort of modification other than the insertion of invariant references, and required by the patentee's limitation of the claims in order to distinguish Okuzumi.

[2] Having set aside Google's actual proposed construction, Red Bend also argues that the phrase "original executable form" in isolation is vague. Red Bend Br. at 25. It is not, and certainly the proposed construction as a whole is clear. *See also* Google Inc.'s Surreply in Support Of Its Opposition To Red Bend's Motion For A Preliminary Injunction at 13 (Dkt. 68).

2. The modified old and new programs/data tables must have "certain portions *replaced*"

a. The specification teaches that the invariant references replace references that change due to delete/insert modifications

Google's full construction—"a version of the actual program or data table in its original executable form, *with certain portions replaced*"—accurately describes what the specification and claims teach. The change that transforms the old and new executable programs/data tables into the modified old and new programs/data tables is the replacement of references that change only due to delete/insert modifications with invariant references. The specification confirms this repeatedly. Ex. B at Abstract, 10:47-50 & 12:11-67; *see also* Google Br. at 18-19. No modification other than the replacement of references with invariant references is hinted. *See Lizardtech, Inc. v. Earth Resource Mapping, Inc.*, 424 F.3d 1336, 1344 (Fed. Cir. 2005) (overly broad construction not supported by specification renders claims invalid under 35 U.S.C. § 112).

Here too, the applicant's statements to the Examiner confirm the correctness of Google's proposed construction. The inventor was at pains to distinguish the claimed invention from the Miller reference, and in doing so disclaimed any modified old or new program/data table that contains auxiliary data such as indexes or tables, rather than replaced references. Google Br. at 19-20. The applicant specifically argued that Miller does not disclose a "modified old program" because Miller teaches the creation of "an index or hash table," which is "auxiliary data [] created in addition to the old file and in contrast to the invention[.]" *Id*. at RedBend0000173. The applicant concluded, "Miller does not disclose the generation of [the] modified old file, and a fortiori, not in the manner recited, for example, in step (a)(I), i.e., 'replacing the reference of said entry by distinct label mark.'" *Id*. at RedBend0000174. Therefore, contrary to Red Bend, the modified old and new programs/data tables cannot be merely any data structure.

b. The Examiner's statements to not affect claim scope.

Red Bend's own description of claim scope is binding. However, Red Bend now argues that the Examiner's statements, and not its own, evidence the proper construction of the claims. That is wrong because the Examiner's unilateral comments do not affect claim scope. *Salazar v.*

Proctor & Gamble Co., 414 F.3d 1342, 1347 (Fed. Cir. 2005) ("examiner's unilateral statements did not alter the scope of the claim"); *see also ACCO Brands, Inc. v. Micro Sec. Devices, Inc.*, 346 F.3d 1075, 1079 (Fed. Cir. 2003). The Examiner observed in his August 23, 2002 Notice of Reasons for Allowance that Miller "creates a modified old program" but does not teach "generating a modified new file." Ex. C at RedBend0000166.[3] In its November 27, 2002 Comments on Statement of Reasons for Allowance, Red Bend disagreed "with the Examiner's contention that Miller creates modified programs *in the sense of the invention*." Ex. C at RedBend0000173 (emphasis added) (also stating "according to Miller, auxiliary data is created in addition to the old file and in contrast to the invention").

In his February 10, 2003 Supplemental Notice of Allowability, the Examiner disagreed with the distinction Red Bend drew between the invention of the '552 patent and Miller. *Id*. at RedBend0000175-77. Red Bend would now rely on the Examiner's statements *disagreeing with Red Bend* to support its proffered claim construction. *See* Red Bend Br. at 20-21 & 24. Red Bend's assertion that the Examiner understood the claims better than it did is obviously strategic.

It is also legally irrelevant. Red Bend's explanation of its claim scope is as informative as it is unequivocal and binding. *Teleflex*, 299 F.3d at 1327; *Phillips*, 415 F.3d at 1317; *Southwall*, 54 F.3d at 1576. That the Examiner may disagree has no bearing on this Court's analysis. *Salazar*, 414 F.3d at 1347; *ACCO Brands*, 346 F.3d at 1079.

3. The unasserted claims do not change the analysis

Red Bend argues that the modified old and new programs need not have replaced references because the asserted claims do not recite the "replacing the reference" language of, for

[3] Contrary to Red Bend, the Examiner did *not* indicate "that a modified old program could be any data structure related to the old program." Red Bend Br. at 21. The portion of Miller cited by the Examiner shows an executable program to which auxiliary data has been appended. *See* Ex. C at RedBend0000166 (referencing Ex. I (Miller) at 6:34-40). The creation of this extraneous data does not alter the program, which remains in its original executable form. Ex. I at 6:34-40 & Fig. 4a. The Examiner stated that *taken together* the old program and the auxiliary data would constitute the modified old program. He nowhere suggested that the auxiliary data alone could constitute the modified old program.

example, unasserted claim 1. Red Bend Br. at 23. But that does not change what the modified old program is. That the asserted claims do not recite the word "replacing" does not change what the nature of the modified old and new programs/data tables. *Laitram*, 143 F.3d at 1463; *ATD Corp.*, 159 F.3d at 541. Moreover, because the non asserted claims include other limitations such as "label mark" and other terms, the doctrine of claim differentiation does not apply. Claim terms are presumed to mean the same thing across claims. *Phillips*, 415 F.3d at 1314. The language of the unasserted claims, when viewed in light of the specification, weigh in favor of construing the term to require that certain portions be replaced.

4. Red Bend's proposed construction would isolate the claims from the specification.

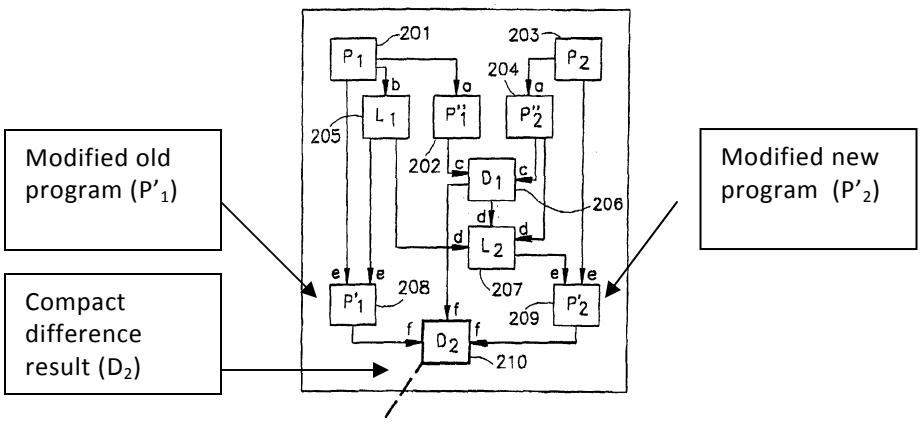
An "interim result" that is "related … *in some fashion*" to the old and new programs/data tables could be practically anything. Red Bend Br. at 21 (emphasis added). By making the modified old and new programs/data tables merely an "interim result," Red Bend would shift the infringement analysis entirely to the starting and ending points of the claimed process, and away from the steps that Peleg described as his invention. Ex. B at claim 8, 9:67-10:15, 10:42-13:4; Ex. E at 11:3-12:23, 20:8-21:9 & 29:15-30:10.

In its effort to justify the "interim result" construction, Red Bend asserts that the modified old and new programs/data tables cannot be accurately identified in the disclosure of the '552 patent. Red Bend Br. at 20.[4] This, the argument goes, means that modified old and new programs/data tables should be construed to cover all of the "interim results" discussed in the specification. *Id.*; Edwards Decl. at ¶ 10. Although the specification includes a single reference to "the various interim results" generated in the process of creating the difference result or in using it to update a program, Ex. B at 9:14-16, the modified old and new programs/data tables cannot be *any* such "interim result."

The specification makes plain that P'_1 and P'_2 in Figures 1 and 2 correspond,

[4] If Red Bend were right, then the claims would be invalid for failure to meet the written description and enablement requirements of § 112, ¶ 1.

respectively, the modified old program/data table and the modified new program/data table. They have to be because the specification calls D_2 "the final difference result." Ex. B at 12:29-30; *see also id*. at claim 8(c) & 13:8-64 (the transmitted "D_2 difference result" is used to create the new executable program, P_2, on the client side). This "final difference result" is generated from P'_1 and P'_2. *Id*. at claim 8(c), 12:35, and Figs. 1 & 2B (showing that the "final difference result" contains "inserted & replaced contends"). Therefore, P'_1 and P'_2 correspond to the "modified old program [data table]" and the "modified new program [data table]" of the claims, just as D2 corresponds to the claimed "compact difference result."



'552 patent, excerpt from Fig. 1 (top portion)

The client-side claims specify that the modified old and new programs/data tables are generated on the client as part of the update process. *See*, *e.g.*, Ex. B at claim 12(b) & (c). The lower portion of Figure 1 depicts this process, and shows P'_1 and P'_2 are involved. Although Red Bend suggests that $P"_1$ and $P"_2$ could correspond to the claimed modified old and new programs/data tables, this is wrong; $P"_1$ and $P"_2$ are *not* depicted in Figure 1 as playing any part in the update process. Ex. B at 13:17-21 & Fig. 1. Red Bend's effort to expand the claims to cover any and every "interim result" based on alleged confusion in the specification is specious.

III. CONCLUSION

For all of these reasons, and those detailed in Google's opening brief, Google respectfully requests that the Court adopt its proposed constructions of the disputed claim terms.

Dated: July 29, 2010

Respectfully Submitted,
Google Inc.,

By its attorneys,

/s/ David M. Magee

Jonathan M. Albano, BBO No. 013850
jonathan.albano@bingham.com
David M. Magee, BBO No. 652399
david.magee@bingham.com
BINGHAM McCUTCHEN LLP
One Federal Street
Boston, MA 02110-1726, U.S.A.
617.951.8000

Susan Baker Manning
susan.manning@bingham.com
Robert C. Bertin
robert.bertin@bingham.com
BINGHAM McCUTCHEN LLP
2020 K Street, NW
Washington, DC 20006-1806
202.373.6000

William F. Abrams
william.abrams@bingham.com
BINGHAM McCUTCHEN LLP
1900 University Avenue
East Palo Alto, CA 94303-2223
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 29, 2010.

/s/ David M. Magee
David M. Magee